Aalto University
School of Engineering
Degree programme in Mechanical Engineering

A Numerical Study of Ice Piece Removal from a Rectangular Tank using a Floodgate

Master's Thesis

27.11.2017

Vaibhav A. Shah

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Technology.

Espoo, November 27, 2017

Supervisor:

Prof. Pentti Kujala

Advisors:

R.U.F von Bock und Polach

Tommi Mikkola

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Vaibhav A. Shah

Title of thesis A numerical study of ice piece removal from a rectangular tank using a floodgate.

Degree programme Master's Programme in Mechanical Engineering

Major/minor Mechanical Engineering **Code** ENG25

Thesis supervisor Prof. Pentti Kujala

Thesis advisor(s) R.U.F von Bock und Polach, Tommi Mikkola

Date 27.11.2017 **Number of pages** 51+11 **Language** English

Abstract

WINICE is a project carried out by researchers at Aalto University and Lappeenranta University. The project aims at investigating wasterwater purification through formation of thin ice layers by natural freezing in regions with cold climate, such that the process is more energy efficient than existing common purification methods. The scope of the project includes purification of wasterwater through formation of thin pure ice sheet in a tank, cutting of the ice sheet into ice pieces, removal of the ice pieces and their transport for further processing. This thesis focuses on a method for collection of the ice pieces after they are cut into pieces.

For the collection process, a floodgate located centrally along the width of the tank is opened to generate an outward flow of water, which causes the ice piece to move along with the water. The ice piece is separated and the water is then is pumped back into the tank for refreezing. The scope of this thesis is studying the feasibility of the removal process, effects of reducing ice sheet length on the time taken to extract the ice piece (reach time) and analysis of reach time as a function of ice piece's initial from the floodgate. The simulation of the ice piece motion is divided in two parts: simulation of water flow under the influence of ice sheet of fixed length, and simulation of a rectangular ice piece motion using the imported water flow field.

The ice piece motion simulation is done for several initial positions using flow fields corresponding to different ice sheet lengths. However, the final simulation flow field from an interpolated case of several cases is needed to account for several distortions in the reach time contours. It is found that the floodgate is capable of extracting ice pieces from even farthest positions in the tank and that most of the time to completely empty the tank, would be spent on removing the ice pieces at the farthest positions from the floodgate. This time can be further reduced by using a wider floodgate.

There are oscillations observed in the reach time vs. distance plots. Regardless, an average trend is noticed for the majority of the domain. The nature of the trendline is a power function of the form $t = Ar^B$ where t is the reach time, r is the distance of the initial position from the floodgate and A,B are constants. Further exploration of the relationship of A and B with different ice piece and flow parameters along with the experimental validation of this simulation is part of the future work.

Keywords ice, extraction, floodgate, tank, OpenFOAM, interFoam, WINICE, wastewater, natural freezing, partial, icesheet

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Prof. Pentti Kujala for giving me an opportunity to work on this topic and be a part of this research. I would also like to thank my instructors, R.U.F von Bock und Polach and Tommi Mikkola, for their kind support and guiding me through this thesis.

I would also like to thank Aalto University for providing me an opportunity to do my masters in Finland and giving me access to a great education and learning opportunities.

Finally, I wish to express my special thanks and love to my mummy (Malti Shah), pappa (Ashwin Shah) and my uncle (Rakesh Shah). Without their continuous support, I would not be able to continue my studies towards a Master's degree.

Espoo, 27th November 2017

Vaibhav Shah

TABLE OF CONTENTS

1 INTRODUCTION 1

1.1 Background 1
1.2 Aim 3
1.3 Thesis outline 4

2 BACKGROUND ON FLOW SIMULATIONS 5

2.1 Overview 5
2.2 Simulating flow using OpenFOAM 5
2.3 Volume of Fluid (VOF) method and interFoam 6

3 OPENFOAM MESH LAYOUT AND PARAMETERS 9

3.1 Shape of the domain and blockMesh 9
3.2 Meshing and Mesh Refinements 12
3.3 Grid Convergence and Cell Parameters 13
3.3.1 Grid convergence for cells in X and Z directions 13
3.4 Assumptions for OpenFOAM simulation 15
3.5 Boundary conditions for OpenFOAM simulation 16
3.6 Prediction of volume flow rate of water 18
3.7 Grid Convergence Simulations, mapFields and results 19
3.7.1 Validity and credibility of mapFields 20
3.8 Grid convergence for cells in Y direction 23

4 SIMULATION OF PARTIAL ICE SHEET CASES 27

4.1 Modelling of ice sheet length variation 27
4.2 Process for running the simulations and exporting results 27

5 SIMULATION OF ICE PIECE MOTION 29

5.1 Assumptions for Implementation in MATLAB 30
5.2 Calculation of Drag Force 31
5.2.1 Calculating average flow velocity at the center of the mass 31
5.3 Calculating Ice-Ice Friction Force 33
5.4 Calculating acceleration, velocity and position of the ice piece 34

6 RESULTS FROM ICE PIECE MOTION SIMULATION 39

6.1 Comparisons between partial ice sheet cases 39
6.2 Feasibility of the Floodgate method 44
6.3 Analysis of reach time v/s distance from the floodgate 46

7 CONCLUSION 48

8 FUTURE WORK 49

REFERENCES 50

APPENDIX 52

MATLAB Code for interpolated ice sheet 52
MATLAB code for individual partial ice case 58

1 INTRODUCTION

1.1 Background

With the ever increasing industrial activity in the world, availability of fresh water has been on decline. The reason for such could be lack of strict environmental rules and or increasing use of industrial processes that produce wastewater as a by-product. One of the key culprits for water shortage and pollution is the mining industry. Large quantities of water is used during mining activities (Mudd et. al, 2008) which causes pollution from acid mine leakage, leakage from tailing, disposal of tailings into water sources, etc. (Akcil et. al, 2006). Some other industries that generate large quantities of wastewater are wood industry, printed circuitborad manufacturing industry and metal electroplating industry (Sörme et. al, 2002). Also, with increase in global population over the past few decades, global consumption of water has increased vastly. This is straining the existing natural water resources. To meet future water requirements for the increasing industrial and domestic consumption, it is essential that wastewater needs to be managed efficiently. One of the important components of wastewater management is the reclamation and purification of clean water from contaminated aqueous solutions.

There are, generally, two kinds of pollutants encountered in wastewater: non-soluble waste and soluble waste. Non-soluble waste is relatively easier to separate from the wastewater by using mechanical separation techniques such as sedimentation, floatation, hydrocycloning and filtration (Lorain et. al, 2001). However, removal of soluble waste can be challenging as the process is often depended on the type of waste dissolved in water. Different processes that are used for wastewater treatment in the industry today include biological processes (not suitable for toxic waste), adsorption (for low concentrations) and oxidation (expensive process for dilute organic compounds). Other methods like evaporation and distillation aim in separating water completely from the dissolved compounds by providing a phase change. However, evaporation and distillation of wastewater can require large quantities of energy. These can be substituted with membrane filtration techniques to improve energy efficiency of the process (Rodriguez et. al, 2000). Membrane filtration processes, however, are suitable for solutes that have a low molecular weight (Lorain et. al, 2001). Also, they require frequent change of filter membranes due to fouling (Kurniawan et. al, 2006) and depending on the type of dissolved waste, these membranes can be expensive and exotic. Even with such diverse methods of purification available, the problem of energy efficiency is not yet solved (Yan et. al, 2011).

Another one of the methods to purify wastewater containing soluble waste is by gradual freezing. This involves purification of wastewater through formation of thin ice layers. Wastewater containing soluble pollutants when frozen gradually, lead to the formation of ice crystals that contain just pure water. The pollutants are rejected into the liquid phase which leads to increased concentration of pollutants in the liquid phase. This happens due to the inability of the ice crystals to incorporate impurities within its matrix (Bogdan et. al, 2014). This is called freeze concentration technique. Lorain et al., studied the effect of freeze concentration on synthetic water (water with one or more soluble pollutants) and industrial wastewater (urban wastewater and cutting oil wastewater). The separation efficiency in most of the cases was found to be close to 100% (Lorain et al., 2001).

Different freezing rates and temperatures will lead to formation of different layered structure of the ice. These ice layers are dependent on the type of the solution, its concentrations and their specific eutectic points. Although, these methods are being studied since past several

decades, the focus has mostly been on using artificial machine cooling to freeze the wastewater and using mechanical scrapers to collect the thus formed ice. In Arctic areas, this use of machines for freezing is unnecessary as the average atmospheric temperature frequently goes subzero in countries like Finland. The freezing process, in theory, can be achieved naturally during these days. Freezing of waste water through natural means can lead to significant energy savings in the purification stage. The amount of energy required for using this technique can be roughly be one sixth the energy needed to evaporate the same wastewater (Mining Magazine, March 2015).

The WINICE Project is a collaboration between researchers from the Department of Chemical Technology at Lappeenranta University of Technology and the Department of Mechanical Engineering (Marine Technology) at Aalto University. The objective of this project is to investigate a novel concept for wastewater purification through formation of ice layers by freeze concentration achieved through natural freezing in cold climate regions. One essential goals for this concept is that the process has to be more energy efficient compared to other existing traditional purification methods. The scope of the project involves purification of wasterwater through formation of thin ice layers, breaking of the newly formed ice layers into ice pieces, collection of the ice pieces and transport of ice pieces for further processing (Mining Magazine, March 2015).

The concept of the WINICE project is shown in Figure 1.1 and consists of several different steps. Each step relies on either a natural energy source or an artificial (secondary) energy source. If the step is completely within the green zone, it means that the energy for that process comes directly from natural sources. Similarly for a step completely in the grey area, the source of energy is completely artificial or secondary. For a step lying between the green and the grey zones, it means that part of the energy comes from artificial sources and other part from nature.

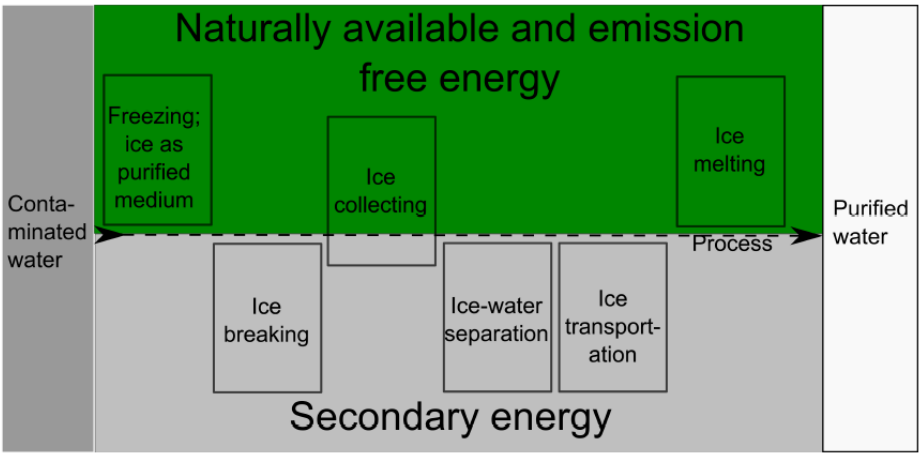

Figure 1. 1: WINICE concept for purification of wastewater using natural freezing and the energy sources for each processes. Figure by R.U.F. von Bock und Polach.

In this concept, the contaminated water is first purified in a tank by gradual natural freezing when the atmospheric temperature is subzero. The gradual freezing of water, due to the subzero atmospheric temperatures, will result in formation of uncontaminated ice layers. Since this process is completely natural, no external energy is needed. The frozen ice (in the form of an ice sheet), is then broken into small pieces, removed from the tank and then collected. The energy required for the ice breaking comes completely from external sources whereas the collecting task is mostly (though not completely) fueled by natural energy

sources. The tasks are designed such that the amount of required external energy is required is kept as low as possible. Finally, the ice pieces are transported to a separate place and are melted naturally by keeping them in a warm place.

1.2 Aim

The method for extracting cut ice pieces, proposed in the WINICE project, is shown in Figure 1.2. After the purification process in a tank (1) containing the wastewater, the ice layer (ice sheet) is broken into thin ice pieces. The floodgate (4), assumed as rectangular in shape is then opened which causes the ice pieces (3) and wastewater to move towards the floodgate and rush out of the tank. The contaminated water is then separated from the ice using a conveyor (6). The separated wastewater is accumulated in the collector (7). This water is then pumped back into the tank from the bottom (5) for another round of purification. This process is repeated until all the ice pieces are collected. The pumping back of the water is what is what requires external energy for the ice collection process in Figure 1.1.

This method relies on external energy sources in some places but the advantage is in the simplicity. It uses the potential energy of the wastewater in the tank to generate flow and move the ice pieces. It eliminates the reliance on external energy sources for the major part of the collection task. The method can also be completely automated and does not limit the portability and the deployment potential of the tank.

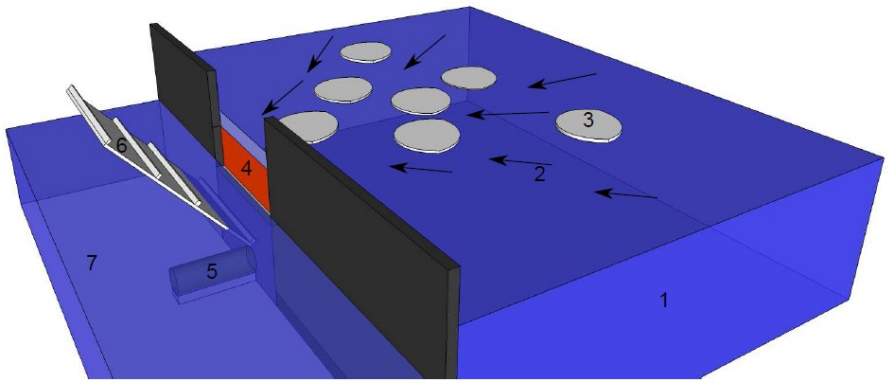

Figure 1. 2: Concept of removing ice and wastewater using floodgate. The concept consists of 1) tank containing wastewater, 2) water flow towards the floodgate carrying ice pieces, 3) ice pieces, 4) the floodgate, 5) pump to pump back the separated wastewater to (1) and 6) conveyor to separate ice from water such that water accumulates in the collector (7). Figure by R.U.F. von Bock und Polach.

Due to these advantages, it is decided to study the feasibility and numerically analyze the removing the thin ice pieces formed due to natural freezing from the aforementioned tank using a floodgate.

The aim of this thesis is to numerically analyze the process of removing thin ice pieces from a rectangular tank using floodgates shown in Figure 1.2.

The thesis covers the following topics:

1. **Simulating motion of a single ice piece in the tank:** As shown in Figure 1.3, opening the floodgate in the tank will induce a water flow due to the potential energy stored in the water. This flow will tend to pull the ice pieces towards the floodgate and thus, out of the tank. In this analysis, motion of a single ice piece in this water flow is simulated.
2. **Effect of different ice sheet lengths on the ice piece motion:** Starting from an ice sheet that covers the entire length and width of the tank, as ice pieces are cut and removed, the length of the ice sheet will progressively decrease until all the ice pieces are removed. Water flow field below the ice sheets of different lengths are compared. This is to find whether the changing ice sheet dimensions have a significant difference in the flow of ice piece.
3. **Feasibility of the method:** The process is deemed feasible if this water flow is able to drag a single ice piece from the extreme end of the tank, in a finite period of time.
4. **Time taken with respect to distance from the floodgate:** For any practical application of this process, one would be interested in the time taken for the removing an ice piece. In the analysis, the time taken for the ice piece to reach the floodgate from its initial cut location (henceforth referred to as reach time) with respect to the distance of the initial cut location from the floodgate, is explored.

1.3 Thesis outline

To analyze the method of removing ice pieces from the tank as per the method in section 1.2, a fluid simulation of the flow of the water in a rectangular tank, due to opening of the floodgate, is done. Based on the simulated flow field of water, approximations are made to calculate the motion of a single ice piece from different initial positions. As the ice pieces are cut and removed from the tank, the dimensions of the ice sheet will decrease. The effect of decreasing dimensions of the ice sheet on the water flow field is studied to find any significant effect of this decreasing dimensions on the flow of ice piece. The time taken by the ice piece to reach the floodgate (reach time) from different starting locations is calculated and plotted. Based on these results, comments are made on the feasibility and the ice piece cutting strategies for this method.

Section 2 in this thesis covers the details of the water flow simulation. It explains briefly the mathematical model used by the third party solver used to simulate water flow.

Section 3 covers in detail the mesh modelling process of the simulation of water flow in the tank and the grid convergence studies. Section 4 discusses about the need to analyze the effect of changing ice sheet length on the reach time of the cut ice piece. It also explains the process of running the OpenFOAM simulations for the partial ice sheet length cases.

Section 5 discusses in detail about the simulation modelling of the ice piece motion based on the water flow field obtained in section 4.

Section 6 presents the results from the ice piece simulation of different partial ice cases and also an interpolated ice sheet length case. This is followed by conclusion, future work and references in the sections 7, 8 and 9 respectively.

2 BACKGROUND ON FLOW SIMULATIONS

2.1 Overview

The problem at hand is similar to a flow of river water through a rectangular weir with a sharp contraction. However, in a traditional weir flow, the water behind the floodgate already has some induced initial velocity which is the river stream velocity. In the problem at hand, the initial induced velocity for water behind the floodgate would be zero as the water is supposed to be at rest in a tank. While looking for existing studies that have similar problems, no such studies could be found from which the flow of water through the floodgate in the rectangular tank could be calculated. Furthermore, no such studies could be found from which flow of thin rectangular ice pieces floating on water can be calculated. This is why, it is decided to simulate this problem using an existing solver.

2.2 Simulating flow using OpenFOAM

The software used for this analysis is OpenFOAM. OpenFOAM (short for Open source Field Operation and Manipulation) is an open source, free to use, toolbox for the development of customized numerical solvers and various pre and post processing utilities to solve problems in the field of continuum mechanics especially computational fluid dynamics (CFD). It is developed by OpenCFD Ltd. The toolbox is written in C++ and has libraries that simplify writing custom solvers. Because of these libraries, that allow researchers to write their own solvers depending on the problem, OpenFOAM has been developing rapidly over the past few years and has generated a huge community following. The toolbox comes with a range of preloaded CFD solvers that can handle a wide range of problems such as single and multiphase laminar and turbulent flows using Reynolds Averaged Navier Stokes (RANS) equations, compressible flows using RANS equations, flows with dynamic meshes, Direct Numerical Simulations (DNS), Lagrangian flows, etc. The toolbox also comes with mesh generation and mesh manipulation utilities that have a friendly to use syntax (OpenFOAM User Guide, 2015).

One of the major disadvantages of OpenFOAM is that it does not come with any graphical user interface and visualization tools. However, it is easy to learn and the visualization of results can be done using 3rd party visualization software like Paraview (used in this case), Blender, VisIt, etc.

The reasons for which OpenFOAM is used for this analysis are: existing knowledge on its know-how, readily available help from the department and from online communities, free of cost and the option of viewing and modifying the source code if needed (open source).

The problem at hand involves simulation of two different phases of fluid: a liquid phase of water and a gaseous phase of air. In Figure 1.3, the blue portion (marked as 1) represents the water in the tank. The surface of this water is in contact with air assuming that the water in the tank is exposed to the environment for freezing. For a solver to calculate accurately the flow field of the water (after the floodgate is opened), it also needs to take into account the motion of the air above water. This is why a two phase fluid solver is necessary.

Along with accounting the two phases of water and air, the solver must also take into account the motion of ice pieces along with the water. Two possible methods to couple the motion of ice pieces together with the flow of water and air are mentioned below:

1. One of the methods to simulate the motion of ice pieces over the free surface of the water, is the use of dynamic meshes. A dynamic mesh is basically a mesh in motion which can be either predefined or a part of the solution. Usually, dynamic meshes are used for relatively a small motion like small disturbance (such as sloshing of water in a tank (Li et. al, 2012) or small displacement of a structural component for generating waves in a tank (Cha et. al, 2011). They work in such a way that the mesh adjacent to the moving object is deformed to accommodate for the movement of the object. However, in this case, the distance that the ice piece have to travel could be large (in the order of several meters) and therefore using dynamic mesh will result in large cell deformations near to the boundary of the solid and the fluid phase. This results in high skewness of cells near moving ice piece which lowers the accuracy of the simulation.

2. Another method to simulate movement of large solid objects in a flow is by overset (chimera) grids. Overset grids is a grid system made up of blocks of overlapping structured grids. A complex geometry is decomposed into a system of geometrically simple overlapping grids. Overset grids require two components from the software side: A preprocessor that generates the overset data (different grid blocks, interpolation weights, etc.) and an equation solver that can understand the overset grid layout. In this case, the mesh would be such that the water and tank part of the domain would form a single set of mesh and the ice pieces floating on the free surface would form a different set. Both meshes will be linked through interpolation. OpenFOAM in its vanilla version does not support the use of overset grids. Although, there exist third party applications that add this feature to OpenFOAM (such as Suggar++), these are costly. There are also several other fluid simulation softwares that are capable of using overset grids. These, also, are proprietary, expensive to purchase and come with their own learning curve.

Owing to these reasons, the simulation of water and ice is decoupled. Instead of simulating the motion of ice pieces directly along with water, the ice piece motion is simulated separately based on the fluid flow simulated using OpenFOAM.

The problem at hand is a multiphase phase flow where water forms one phase, air the second and ice is the third phase. And since the simulation of ice and water is decoupled, the total volume of fluid inside the simulation domain will be composed of only two phases, air and water. OpenFOAM comes with a preinstalled solver for two phase laminar and turbulent flows called "interFoam". This solver works on the volume of fluid (VOF) method phase fraction interface capture method. The solver can be used to compute the flow for two incompressible, isothermal, immiscible fluids (OpenFOAM User Guide, 2015).

2.3 Volume of Fluid (VOF) method and interFoam

The Volume of Fluid (VOF) method is a numerical technique that is based on the marker and cell techniques used to model, track and locate free surface or fluid-fluid interface. It is an advection scheme (numerical scheme that allows user to track the position and shape of the interface) that is dependent on Navier-Stokes equations for describing the motion of the flow. It was first presented by Hirt and Nichols (Hirt and Nichols, 1981).

The method uses a single variable called phase fraction to describe the fluid state in a cell. α is a function used to define the state of fluid fraction in a cell. It is a scalar function that represents the volume fraction of a fluid in a computational cell. This volume fraction of is tracked for each cell in the computational grid.

When a cell is empty such that there is no fluid inside, the value of α is set to zero. Similarly, when the cell is completely occupied by the fluid, the value of α is set to unity. For a cell that does not lie at the fluid-fluid interface, the value is either zero or one. If the value of α lies between zero and one, it means that the cell is part of the fluid interface. It is a discontinuous function and its value jumps from zero to one when moved away from the fluid interface towards the interior of the fluid. The normal direction of the fluid interface is found where the value of α changes the most rapidly. The free surface in this method is defined roughly and is distributed over the height of the cell. To achieve high accuracy and high resolution of the fluid interface, local refinements such that only cells with α between zero and one are refined. For a case with more than two fluids, the sum of α corresponding to each fluid in a cell is one. In the case of a two phase simulation, if α represents the volume fraction of one fluid, then (1- α) represents the volume fraction of the other.

The method in which this method is implement in OpenFOAM is described extensively by Ubbink (Ubbink O., 1997 and Ubbink H., 2002). A more concise description is presented by Berberović (Berberović, 2009).

In the conventional VOF method, the transport equation (Equation 2.2) is solved simultaneously for an indicator function along with the continuity (Equation 2.1) and momentum (Equation 2.3) equations. The transport function represents the volume fraction of one phase.

$$\boldsymbol{\nabla . U = 0} \tag{2.1}$$

$$\frac{\boldsymbol{\partial \alpha}}{\boldsymbol{\partial t}} + \boldsymbol{\nabla .(U\alpha) = 0} \tag{2.2}$$

$$\frac{\boldsymbol{\partial (\rho U)}}{\boldsymbol{\partial t}} + \boldsymbol{\nabla .(\rho UU) = -\nabla p + \nabla .T + \rho f_b} \tag{2.3}$$

U is velocity field shared by the two fluids throughout the flow domain, α is the the phase fraction, T is the viscous stress tensor, ρ is the density, p is the pressure and f_b is the body forces per unit mass.

In VOF method, the body forces include gravity and surface tension at the interface. The phase fraction α can be in the range [0,1] with zero corresponding to no fluid in the cell and one corresponding to the cell fully occupied with the fluid. For example, for a gas, $\alpha = 0$ and for a liquid $\alpha = 1$. Only in the region of the interface are the gradients of the phase fraction α are encountered.

Two immiscible fluids are considered as one fluid throughout the entire domain. The physical properties are calculated by the weighted averages based on liquid volume fraction. Thus, the properties of the fluid in their corresponding occupied regions are given by:

$$\boldsymbol{\rho = \rho_l \alpha + \rho_g (1 - \alpha)} \tag{2.4}$$

$$\boldsymbol{\mu = \mu_l \alpha + \mu_g (1 - \alpha)} \tag{2.5}$$

Where, ρ and μ are density and dynamic viscosity respectively and subscripts l and g stand for liquid and gas respectively.

In present day, a modified approach similar to (Ubbink H., 2002) is used by OpenCFD Ltd. for interFoam. It relies on two-fluid formulation of conventional VOF model in the framework of finite volume method (Rusche H., 2002). An additional convective term is introduced into the transport equation of the phase fraction (Equation 2.2). This term

originates from modeling the velocity in terms of the weighted average of the corresponding liquid and gas velocities. This provides a sharper interface resolution. The model uses two-fluid Eulerian model for two phase flow. The phase fraction equations are solved separately for each individual phase. The equation for each phase fraction are:

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla}.(\boldsymbol{U_l \alpha}) = \mathbf{0} \tag{2.1}$$

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla}.\left[\boldsymbol{U_g}(\mathbf{1}-\boldsymbol{\alpha})\right] = \mathbf{0} \tag{2.2}$$

The subscripts l and g stand for liquid and gaseous phase respectively. It is assumed that the contributions of the liquid and gas velocities to the development of free surface are proportional to the corresponding phase fraction. The velocity of the effective fluid in the VOF is modeled as the weighted average.

$$\boldsymbol{U} = \boldsymbol{U_l \alpha} + \boldsymbol{U_g}(\mathbf{1}-\boldsymbol{\alpha}) \tag{2.3}$$

Equation 2.6 can be rearranged and used with equation 1.8 to obtain the evolution of the phase fraction α.

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla}.(\boldsymbol{U\alpha}) + \boldsymbol{\nabla}.[\boldsymbol{U_r \alpha}(\mathbf{1}-\boldsymbol{\alpha})] = \mathbf{0} \tag{2.9}$$

Where, $U_r = U_l - U_g$ is the relative velocity vector and is called as the "compression velocity".

The above equation 2.9 contains an additional convective term referred to as the compression term. The role of this term is to "compress" the free surface towards a sharper one. The additional convective term contributes to high interface resolution. This term is valid only for the interface region and disappears at both limits (0 and 1) of the phase fraction.

VOF is a light computationally since it only uses one parameter to keep track of the fluid-fluid interfaces. However, the resolution of the interface using the conventional VOF method is not so accurate. This is usually overcome by using a two-phase fluid coupling (in case of two-phase cases) as shown above for the interFoam solver.

Thus, the interFoam solver in the OpenFOAM is a good choice for the water simulation in this case. The following section deals with the mesh modelling and finding out optimum parameters for the water flow simulation using the interFoam solver.

3 OPENFOAM MESH LAYOUT AND PARAMETERS

In Section 2, it is explained that the simulations are separated into two parts such that the first part is the simulation of water flow using OpenFOAM and the second part is the simulation of ice piece motion using the flow field generated by OpenFOAM. One of the most important aspects of fluid simulations is mesh modelling.

3.1 Shape of the domain and blockMesh

For the simulation of water flow in OpenFOAM, a 3D rectangular tank of is constructed using the blockMesh utility. blockMesh is a meshing utility that comes prepackaged with OpenFOAM. The utility uses blocks of cells (typically, but not necessarily, hexahedron) laid on top and or next to each other to make up the domain. The number of cells and the grading (type and size of the cells) can be controlled independently for each block. The cells of two adjacent blocks share common faces with each other. To maintain continuity between two blocks, the cell parameters have to be such that the number of cells and grading between two blocks are same for directions in which the two blocks share a common face.

For this analysis, the tank is assumed to be of a rectangular shape. The dimensions of the tank in the mesh are 10m x 10m x 2m in length (X), height (Y) and width (Z) respectively with the height of the partition wall (hatched area in Figure 3.1) as 1m. Figure 3.1 shows how different hexahedron blocks are stacked on top of each other in 3D to make the block mesh. Henceforth, the axes X, Y and Z are used for lengthwise, heightwise and widthwise directions, respectively, in the tank.

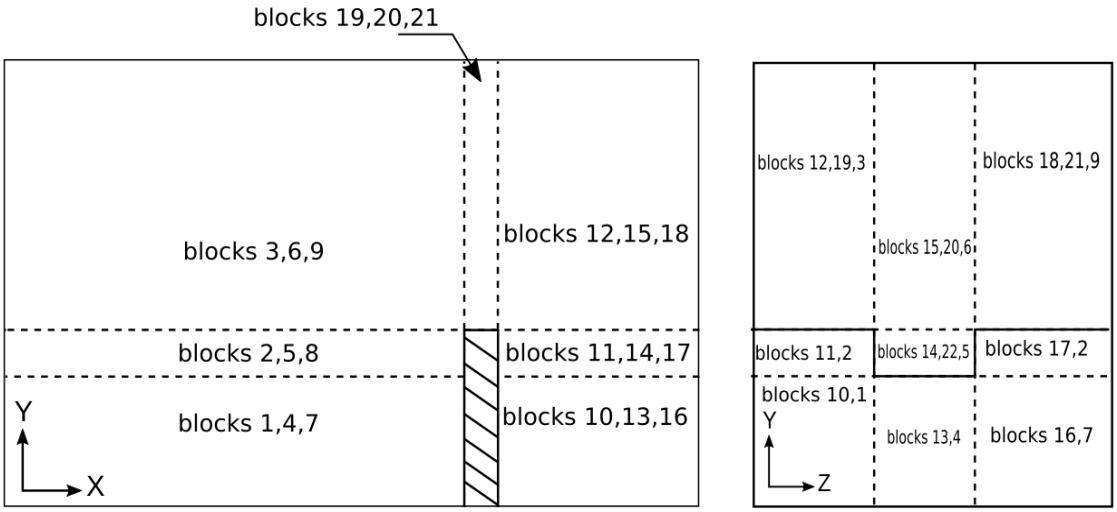

Figure 3. 1: Block layout and numbering used for blockMesh utility. The hatched section represents the partition wall. X, Y and Z represent the direction of length, height and width (for all figures henceforth).

The tank (blocks 1, 4, 7, 2, 5 and 8 in Figure 3.1) is separated from the collector (blocks 10, 13, 16, 11, 14 and 17 in Figure 3.1) by a partition wall (shown as cross-hatch in Figure 3.1) which has an opening for the floodgate. The floodgate is represented by blocks 14, 22 and 5 in the Figure 3.1 (right image). The collector is the part where the water flows out to, from the tank, through the floodgate. For this analysis, the collector part merely exists to separate the tank from the outlet.

The length of the collector is taken to be 4m in the mesh. This is to keep sufficient distance between the floodgate and the end wall so that the end wall (collector wall parallel to the partition wall) does not interfere with the water outflow. The width and height of the collector are the same as the tank side.

The mesh domain is modelled such that the floodgate is at the center of the tank width (Z direction) as shown in Figure 3.3. A quick crude simulation done using a coarse mesh shows that the velocity profile of the water flow in the tank is symmetrical about a plane parallel to the X-Y plane, passing through the center of the floodgate with respect to the width (Z = 5). Figure 3.2 shows a cross section of the crude flow field parallel to the X-Z plane and passing through Y = 1.9 (center height of the floodgate). The velocity contours for 0.5, 0.25, 0.1, 0.05 and 0.025 m/s, respectively, are shown by white curves. The contours are symmetrical in the X-Z plane since the floodgate is located at the center of the tank widthwise. The plane of symmetry (black dotted line) passes through the center of the floodgate parallel to the X-Y plane. Due to this symmetric nature of the flow, it would be sufficient to simulate the water flow for just one half of the mesh rather than simulating it for the entire mesh.

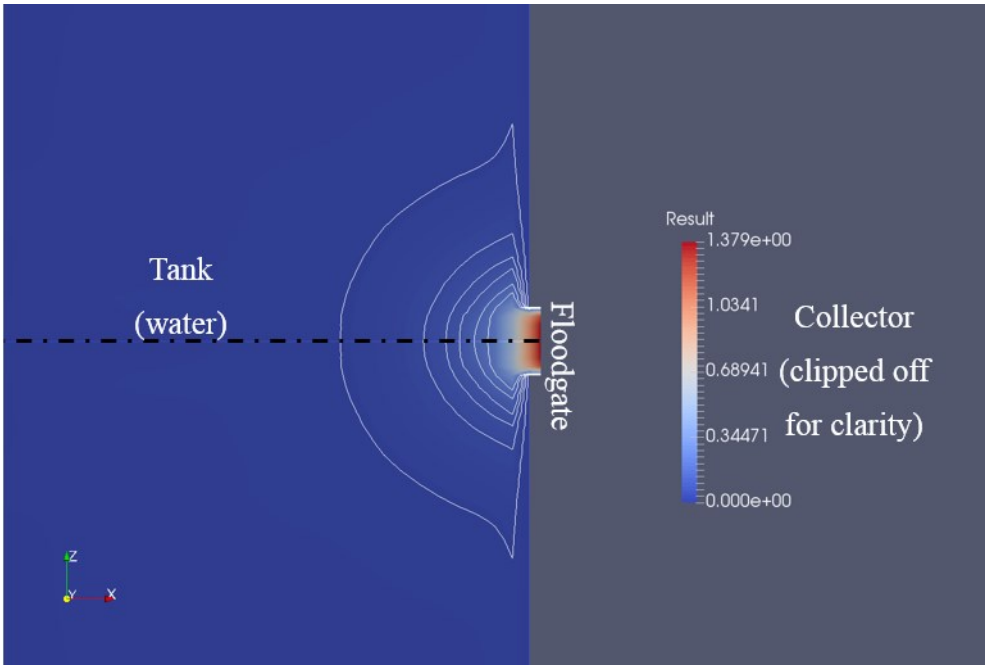

Figure 3. 2: Flow symmetry in the X-Z plane. Plane of symmetry passing through the center of the floodgate parallel to the XY plane. Here the term results represents the magnitude of flow velocity. The contour closest to the floodgate is of 0.5 m/s. The velocities of the subsequent contours away from the floodgate are 0.25, 0.1, 0.05 and 0.025 m/s respectively.

A symmetry plane parallel to the XY plane and passing through the center of the floodgate width at Z = 5, is used to model the flow symmetry. The symmetry plane reduces the total number of cells in the mesh by a factor of 2. The number of operations performed per iteration in a simulation in roughly proportional to the number of time steps and the cube of the number of points in the mesh (Deville and Gatski, 2012). By halving the total number of cells in the mesh (thus, the number of points in the mesh), the total simulation runtime for the same number of time steps is reduced by a factor of 8. Figure 3.3 shows a line diagram of the mesh without symmetry and Figure 3.4 shows that with symmetry plane.

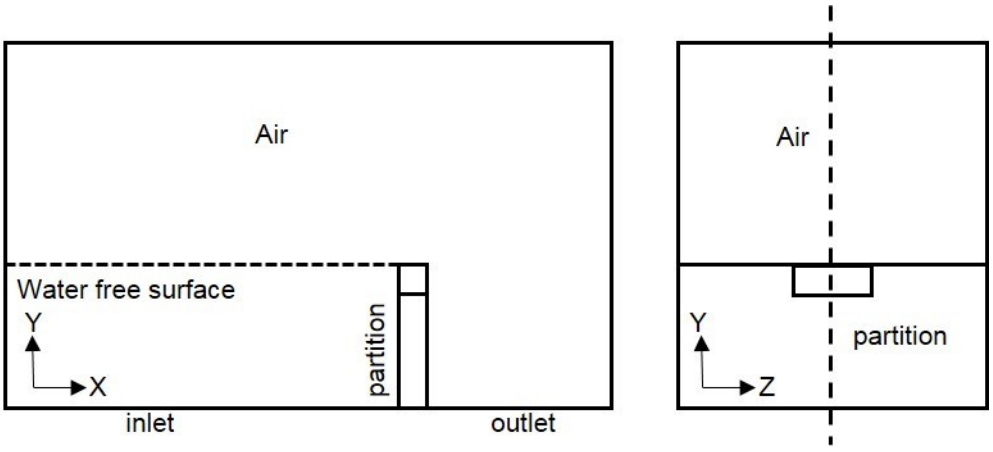

Figure 3. 3: Line diagram of the mesh without using symmetry

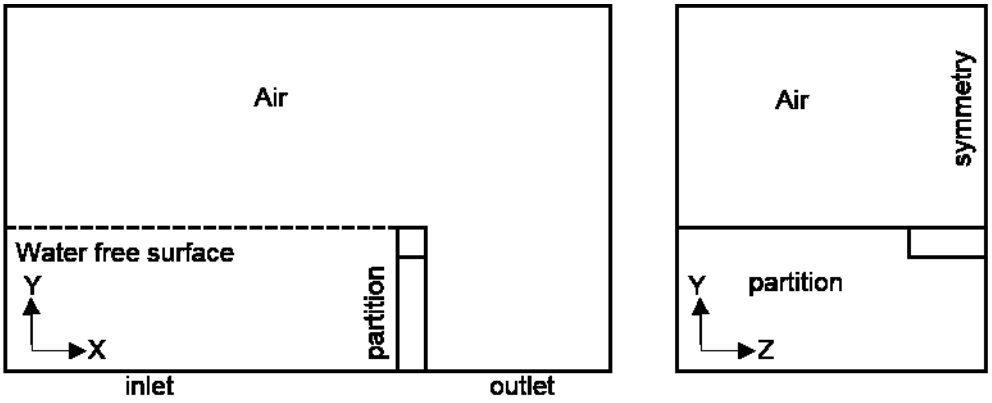

Figure 3. 4: Line diagram of the mesh with symmetry

In Figure 3.4, the water surface is exposed to the atmosphere meaning that there is no ice sheet modelled in the mesh. In section 2.2, it is concluded that the simulation of ice piece motion is decoupled from the water flow. The ice piece motion is calculated based on the imported water flow field in OpenFOAM. For this purpose, it is essential to know the development of velocity profile of the water near the ice. Also, the starting location of the ice piece can be anywhere in the tank, an ice sheet needs to be modelled for the final simulations. One of the aims of this analysis is to study the effect of changing ice sheet dimensions, the mesh is modelled in such a way that the length of the ice sheet can be changed easily. The ice sheet is modelled using the createPatch utility available in OpenFOAM. It is a utility that allows users to create patches from selected mesh cells/faces.

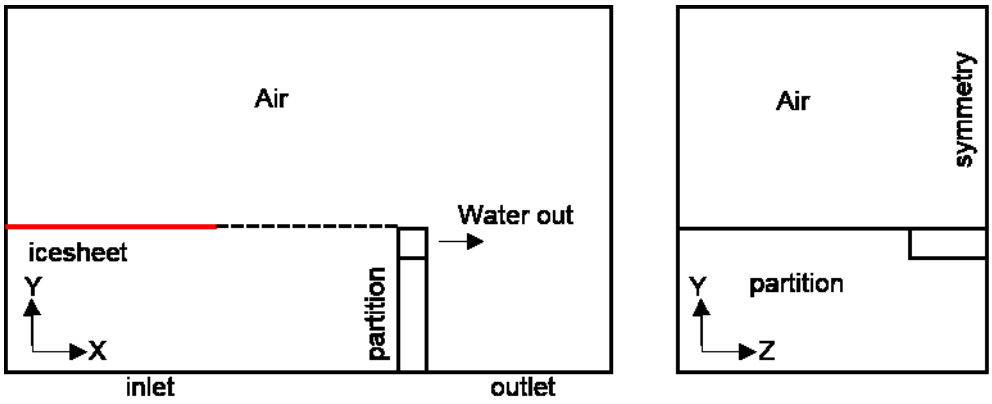

Figure 3. 5: Introduction of the ice sheet (red line) to account for the boundary layer on the ice bottom face.

To summarize, the final shape of the mesh domain is changed from the actual physical case (in Figure 3.3) to a domain with symmetry conditions to reduce simulation run time. Also, an ice sheet shown in Figure 3.5 is modelled in order to get the capture the boundary layer generation near the ice piece bottom face. The length of the ice sheet is variable in order to study the effect of changing ice sheet dimensions.

3.2 Meshing and Mesh Refinements

The mesh for OpenFOAM simulation is generated in two steps. The first step involves making a base mesh using the blockMesh utility as explained in the beginning of this section. The ratio of the sizes of the adjacent cells in all directions is kept under 1.2. Keeping the cell size ratio below 1.2 is considered to be a good practice while designing meshes in OpenFOAM to reduce numerical errors (Prof. Timo Siikonen).

The second step involves using mesh refinement tools to increase the number of cells near the floodgate. The reason for having more cells near the floodgate is to accurately capture the high velocity and pressure gradients near the floodgate caused by the large difference in the hydrostatic pressure on two sides (tank side and collector side) of the floodgate, when the floodgate is opened. To capture these gradients accurately, a finer mesh is required near the floodgate. The gradients become progressively smaller, away from the floodgate as a result of which, the mesh becomes progressively coarser away from the floodgate.

The mesh refinements in this analysis are achieved using the refineMesh utility that comes prepackaged with OpenFOAM. The refineMesh utility takes the mesh generated by the blockMesh utility, as a base. The user is able to select a region from the blockMesh generated mesh and refine the cells in just that region. The refinements can be done in X, Y or Z directions. It can also be in combinations of the three directions. The term refinement in this context means that a single cell is further split into two cells in a given direction. If refinements are done in all the three directions, a single 3-D cell is broken down (split) into 8 smaller cells. Refinements increase the cell count in the selected refinement region by a factor of 2^n where n in the number of directions in which refinements are done. For refinements in only one axis, the cell count in the selected region is doubled. Another round of refinement in the same region will further break the already broken cells in the same fashion. By this process, the number of cells in a particular region can be increased significantly without affecting other regions. The user can make the required areas finer and have control over the mesh quality in the desired regions. However, it should be noted that

the total number of cells in the mesh can go up drastically which can significantly increase the computation time. It is always desirable to achieve a balance between the desired accuracy and the total cell count (computation time). For this, a grid convergence study is necessary. In a grid convergence study, meshes of different cell densities and cell counts are used to run the same simulation and the results from each cases are compared. The cell count is increased till the gains in accuracy is either zero or insignificant. In the end, cell parameters from the case which gives this balance of sufficient accuracy and computation time, are used to perform the final simulations.

More about mesh refinements is explained in Section 3.3.1. However, a point to note here is that the refinements are only done in the X and Z directions with most refinements being near the floodgate.

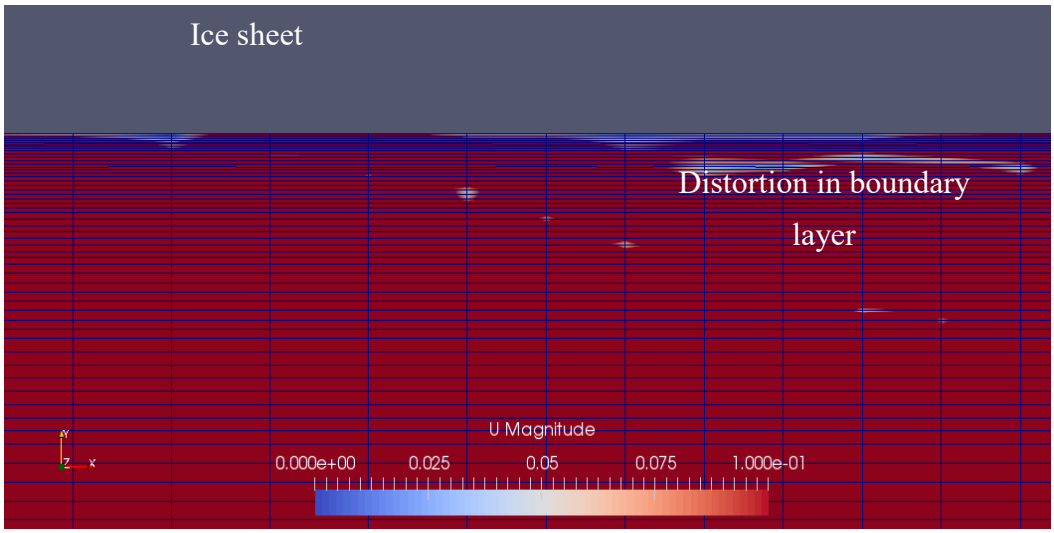

Figure. 3. 6: Distortion in the boundary layer due to mesh refinement in the Y direction. U represents the velocity in the X direction.

Refinements done in Y direction result in the distortion of the boundary layer (an important feature needed for the ice piece motion) near the ice sheet as shown in Figure 3.6. For this reason, the cell parameters in Y direction are determined from a separate study.

3.3 Grid Convergence and Cell Parameters

To have a balance between computation time and reliable accuracy, grid convergence studies are done to determine the cell parameters that give a balance between required accuracy and computation time. Because the refinements in Y direction distort the boundary layer (Figure 3.6), the grid convergence is separated into two phases:

1. Grid convergence for cells in X and Z directions without any ice sheet.
2. Grid convergence in Y direction with a mesh containing ice sheet

3.3.1 Grid convergence for cells in X and Z directions

This test is performed using a mesh with a symmetry plane but without an ice sheet.

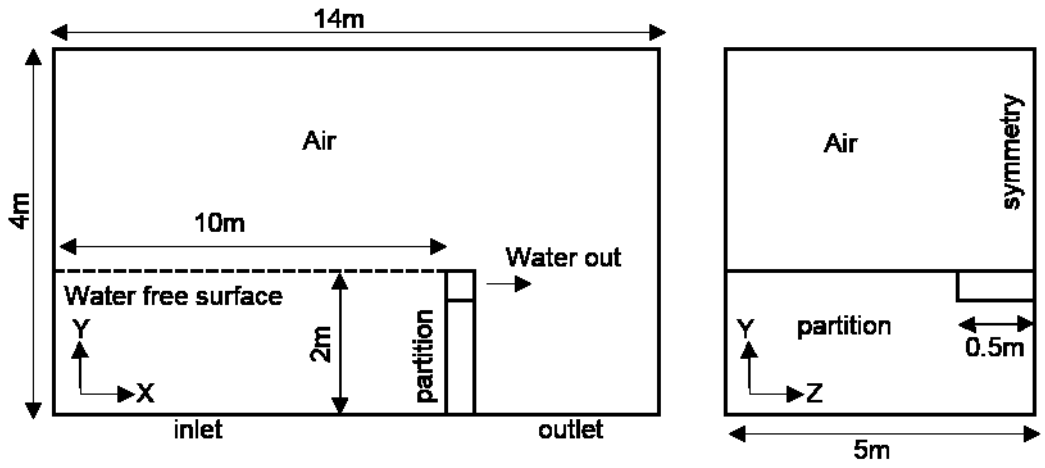

Figure 3. 7: Line diagram for the mesh used for X-Z grid convergence.

The dimensions of the tank are 10m, 5m (after symmetry plane) and 2m in length, width and height respectively. The bottom face of the tank is the inlet through which the removed water is pumped back. The width of the floodgate is 0.5m. For all subsequent simulations, the floodgate is assumed to be of 0.5m width after the symmetry condition is applied.

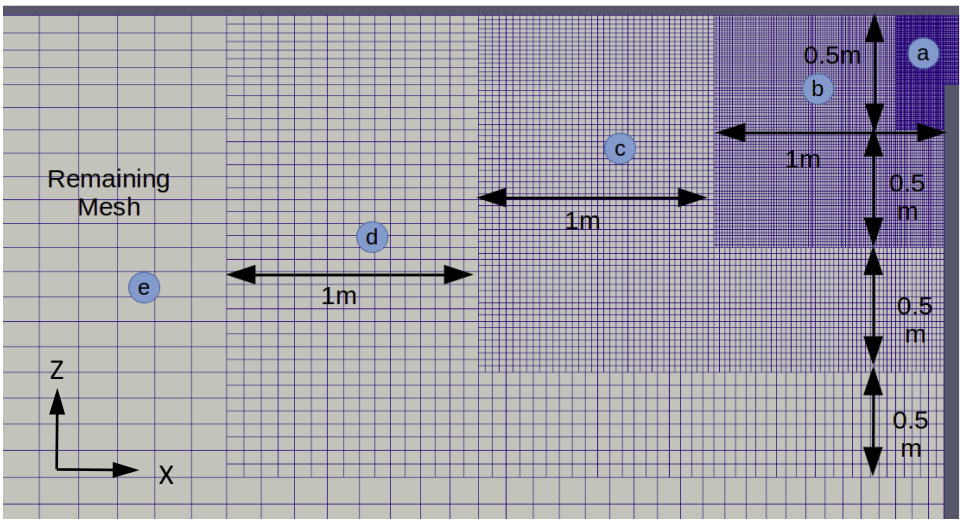

Figure 3. 8: Refinement zones used for the refineMesh utility

Figure 3.8 shows the cell refinements on a cross section of the tank side in the X-Z plane at a height of Y = 1.9 (center of the floodgate height). The floodgate is located within the region *a*. The length (X direction) of each zone is 1m more than the previous zone. The length of zone *b* is 1m, that of zone *c* is 2m and similarly for other subsequent zones. The length of zone *a* before the floodgate is 0.2m. Zone *a* extends from the tank side to the collector side through the floodgate to ensure that there are no major discontinuities between the cell sizes in tank side (near the floodgate), inside the floodgate and in the collector side. This continuity in cell size prevents any large numerical errors in this region (of high pressure and velocity gradients) of the mesh. Similar to the length of the zones, the width of each zone is 0.5m more than the previous zone. The width of zone *a* is 0.5m. That of zone *b* is 1m and similarly for other subsequent zones. The number of refinements applied decreases from zone *a* to *e*. Number of refinements applied in zone *a* is 4. This number decreases by 1 for each subsequent zone such that zone *b* has 3, zone *c* has 2, zone *d* has 1 and zone *e* has no refinements.

Four separate test cases (meshes) are prepared such that the ratio of the total number of cells between two subsequent cases is approximately 1.5. For each test case, for each refinement zone, minimum and maximum cell sizes in X and Z direction are tabulated in Table 1.

Table 1: Cell parameters for grid convergence in X and Z directions

1	**Total cell count**	**Refinement region**	**xmin (mm)**	**xmax (mm)**	**zmin (mm)**	**zmax (mm)**
1	232148	a	0.0064	0.0095	0.0047	0.0068
		b	0.018	0.022	0.0094	0.0163
		c	0.037	0.052	0.02	0.038
		d	0.0735	0.125	0.0375	0.085
		e	0.15	0.5	0.075	0.35
2	355959	a	0.0063	0.00875	0.00375	0.00562
		b	0.017	0.0206	0.0075	0.0134
		c	0.034	0.05	0.015	0.031
		d	0.06	0.114	0.03	0.0712
		e	0.135	0.45	0.06	0.28
3	504490	a	0.0064	0.00789	0.0027	0.0048
		b	0.0188	0.0153	0.0067	0.0117
		c	0.0306	0.0436	0.0125	0.0265
		d	0.0612	0.104	0.022	0.619
		e	0.123	0.408	0.051	0.244
4	752636	a	0.00418	0.0044	0.002	0.0035
		b	0.00835	0.0116	0.0058	0.0109
		c	0.0167	0.0296	0.0188	0.022
		d	0.0334	0.0737	0.019	0.0577
		e	0.0668	0.334	0.043	0.162

3.4 Assumptions for OpenFOAM simulation

To reduce complexities of the problem, a few assumptions are made for the OpenFOAM simulation. The assumptions made for all OpenFOAM simulations are as follows:

1. **Flow is laminar:** The range of velocities induced by the floodgate are relatively small even near to the floodgate. From running coarse simulations without any inlet from the

bottom wall, it is observed that the maximum flow velocities, which occur usually very close to the floodgate, are in the range of 1-2 m/s. The flow velocities decrease rapidly away from the floodgate. As a result, a transient is developed near the floodgate at the start of the simulation. More about this transient is discussed in section 3.7. Eventually, this transient dampens and the flow becomes laminar. Thus, it is safe to assume the general flow to be laminar in nature.

2. **Properties of water:** For an ice sheet to develop, the atmospheric temperature has to be subzero. For this analysis, the atmospheric conditions are uncertain which makes it difficult to predict the temperature of the water and thus its properties. Therefore, an average of the density and viscosity values within the range of 0 to 4 °C is taken as in input to the OpenFOAM solver. The average values of density and kinematic viscosity between 0 to 4 °C are 1000 kg/m^3 and 1.656E-06 m^2/s respectively (Kestin et. al, 1978). These values are part of the initial conditions for the interFoam solver.
3. **No loss of water:** In the method defined in Figure 1.2, the water flowing out from the floodgate is immediately pumped back into the tank once the ice pieces are separated from it. It is, therefore, assumed that no water is lost or wasted during this process. Effectively, the water inflow rate into the tank is equal to the water outflow rate through the floodgate.
4. **Quasi-steady state after considerable time steps:** Since the rate of water inflow rate in the tank is the same as the outflow rate through the floodgate, it is assumed that, after a certain number of time steps, the simulation would reach an approximate steady state (equilibrium) and becomes time independent. Once it reaches this quasi-steady state, the difference between the flow field at that time step and the previous time step is negligibly small.

3.5 Boundary conditions for OpenFOAM simulation

Figure 3.9 shows the names of the faces/patches used by OpenFOAM to refer the mesh boundaries. The boundary conditions used by the interFoam solver in OpenFOAM for pressure (p_rgh file), velocity (U file) and the alpha value (volume fraction parameter in Section 2) for water (alpha.water file) are mentioned in Table 2. Along with the name of the boundary condition used, Table 2 also mentions the values of the certain necessary parameters for the respective boundary conditions. For grid convergence study in X and Z directions, the length of the ice sheet is 0. This effectively means that the tank has no ice sheet.

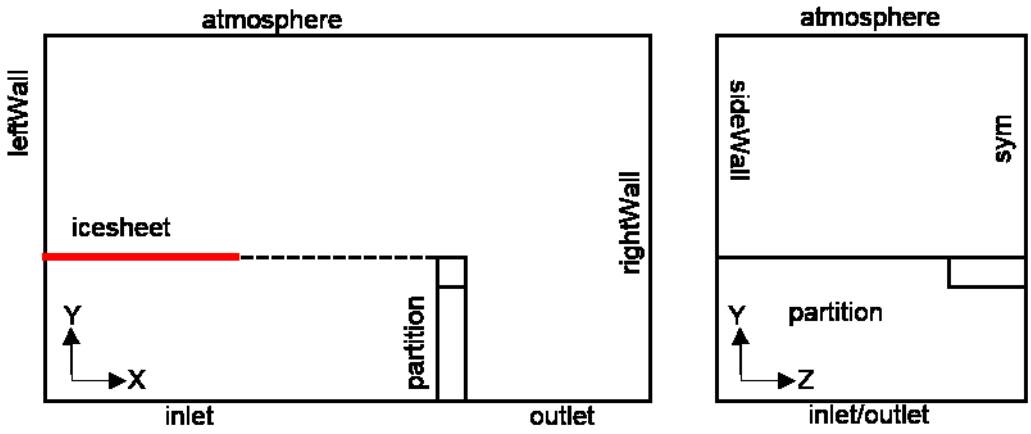

Figure 3. 9: Face/Patch names used in OpenFOAM to identify different faces in the mesh

Table 2: Boundary conditions for velocity, pressure and alpha.water used in OpenFOAM for patches corresponding to Figure 3.9.

Patch	**U**	**p_rgh**	**alpha.water**
leftWall	Type: fixedValue Value: uniform (0 0 0)	Type: zeroGradient	Type: zeroGradient
rightWall	Type: fixedValue Value: uniform (0 0 0)	Type: zeroGradient	Type: zeroGradient
sideWall	Type: fixedValue Value: uniform (0 0 0)	Type: zeroGradient	Type: zeroGradient
partition	Type: fixedValue Value: uniform (0 0 0)	Type: zeroGradient	Type: zeroGradient
atmoshpere	Type: pressureInletOutletVelocity Value: uniform (0 0 0)	Type: totalPressure P0: uniform 0 U: U phi: phi rho: rho psi: none gamma: 1 Value: uniform 0	Type: inletOutlet inletValue: uniform 0 Value: uniform 0
inlet	Type: variableHeightFlowRateInletVelocity flowRate: flow rate for the case Alpha: alpha.water Value: uniform (0 0 0)	Type: zeroGradient	Type: variableHeightFlowRate lowerBound: 0 upperBound: 1 value: uniform 0

outlet	Type: pressureInletOutletVelocity Value: uniform (0 0 0)	Type: totalPressure P0: uniform 0 U: U phi: phi rho: rho psi: none gamma: 1 Value: uniform 0	Type: inletOutlet inletValue: uniform 0 value: uniform 0
sym	Type: symmetry	Type: symmetry	Type: symmetry

3.6 Prediction of volume flow rate of water

One of the assumptions for the OpenFOAM simulations is that the outflow rate of water through the floodgate is equal to the inflow rate into the tank. Since the average outflow velocity through the floodgate is unknown prior to the simulation, it is difficult to calculate the flow rate of the water through the floodgate. When running a coarse simulation of the above mesh without any inlet conditions, the maximum velocity of water near the floodgate is found in the range of 1 m/s to 1.5 m/s. An estimate of the flow rate is calculated using this knowledge. The average flow velocity through the floodgate is assumed as 1.5 m/s (upper limit of the range). An initial volume flow rate is calculated by multiplying the cross section area of the floodgate with this assumed average velocity. This flow rate is then taken as a first value for the flowRate parameter in Table 2. A coarse test mesh which is basically the base mesh generated from the blockMesh without any refinements (to lower total cell count), is generated.

A simulation for this test mesh is run till the time step of 50 seconds and the flow field and water level is visualized on a XZ cross section at Y = 2.0. Assuming the average velocity as the upper limit of the velocity range (1.5 m/s) ensures that the first flow rate is always overestimated and the correct flow rate is approached from higher side by reducing the flowRate parameter by 0.005 and repeating the simulation until the water level starts decreasing (Figure 3.10, left image). The correct flow rate for that floodgate width is considered to be $\emptyset + 0.005$ m^3/s which is the flowRate value for previous iteration. This provides a very close approximation of the flow rate of water for different floodgate width.

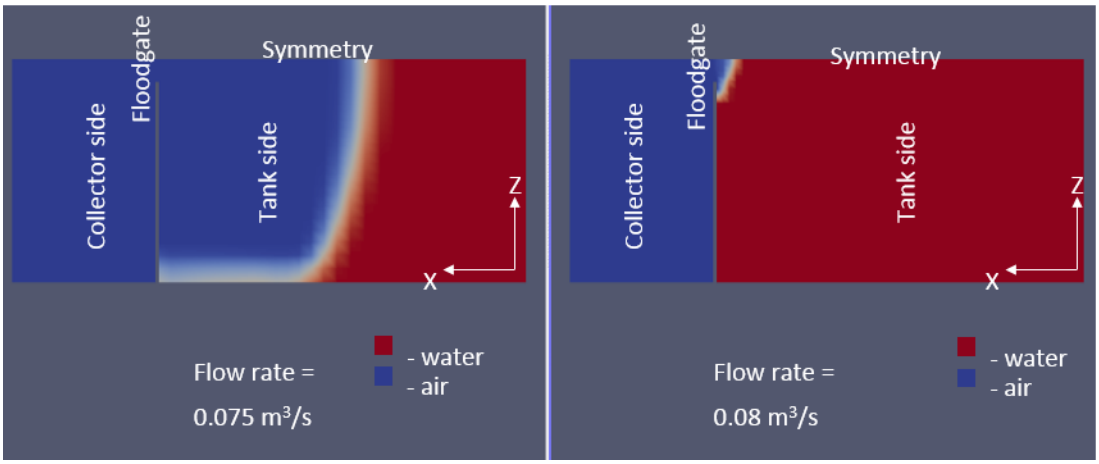

Figure 3. 10: XZ cross section at Y=2.0. On left, the flow in the tank is not equal to flow outside the tank and on right flow in the tank is equal to the flow outside the tank. Red represents the water in the tank blue represents the air. Floodgate width is 0.5m.

In Figure 3.10, the tank is viewed from the top. In the left picture, the water entering the tank (0.0075 m^3/s) is not enough to replace the water going out. Thus, the difference in the volume is replaced by the air (blue part). On the right picture, the water entering the tank is approximately enough to replace the water leaving the tank through the floodgate and thus the water remains at a steady height. Thus 0.08 m^3/s is a good approximation for flow rate in this case.

3.7 Grid Convergence Simulations, mapFields and results

After defining the boundary conditions, the simulation is run for a final time step of 10 seconds. In this, it is observed that when the floodgate is just opened, a wave (disturbance) originates at the floodgate due to the huge difference in hydrostatic pressure between the two sides (tank and collector) of the floodgate (Figure 3.11).

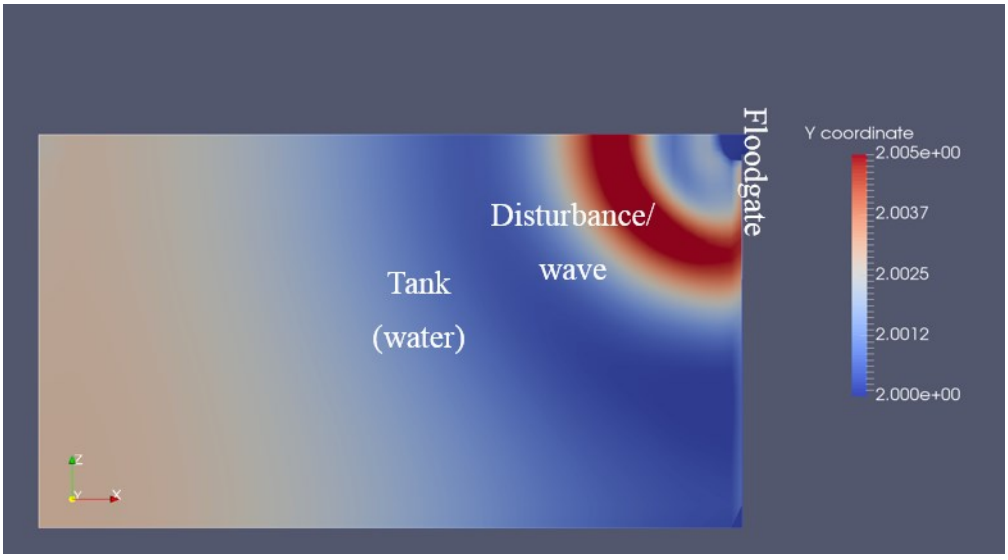

Figure 3. 11: Disturbance origination near the floodgate 2 seconds after the floodgate is opened.

This wave then propagates behind the floodgate towards the tank walls, collides with the tank walls and gets reflected towards the floodgate. These reflected waves from X and Z directions interfere in the tank and cause standing waves. It is necessary that these standing waves are dampened out so that the flow becomes steady, before the flow fields from the different cases in Table 1 are compared. A steady flow is necessary for an accurate and reliable comparison between the different mesh cases. If the simulations are run for considerable time steps, these waves dampen out due to the viscosity of the water. However, this would considerably increase the total computation time.

To combat this increase in computation time, OpenFOAM's mapFields utility is used. This utility allows mapping data from an already computed source geometry to "to be computed" destination geometry. Flow field from an already simulated coarse mesh can be used as the initial condition for the actual simulation mesh. A simulation is run on a relatively coarse mesh from the time step 0 until a time step, t_1, such that t_1 is less than the final time step (t). Using mapFields, flow field from this coarse mesh, at time step t_1, is mapped on the final (actual) simulation mesh as the initial condition. The final simulation is then run with the finer mesh, starting from time step t_1 to the final time step t. The simulation on the finer mesh is only run for a shorter time period of t_1 to t (as opposed to 0 to t without mapFields), which reduces the total simulation time.

The various cell parameters, used for the mapFields coarse mesh, are mentioned in Table 3. Simulation on this mesh is run from time step 0 to 250s. At this point, it is found that the waves have dampened out and the flow field is steady. Using mapFields, the flow field from the coarse case in Table 3 at time step 250s is mapped on to the cases mentioned in Table 1. The mapped flow field is the initial condition for the simulation on the final meshes in Table 1. Simulation final meshes are then run from the time step 250s to 300s.

Table 3: Cell parameters for the coarse mesh used for mapFields

Total cell count	**Refinement region**	**xmin (mm)**	**xmax (mm)**	**zmin (mm)**	**zmax (mm)**
152280	a	0.0065	0.00951	0.00625	0.00924
	b	0.0184	0.0219	0.0125	0.0221
	c	0.0367	0.0523	0.025	0.05
	d	0.0735	0.125	0.05	0.113
	e	0.147	0.49	0.1	0.464

3.7.1 Validity and credibility of mapFields

The reliability of the mapFields utility to reduce simulation time while maintaining accuracy is evaluated by comparing a case run using results from mapFields as initial conditions (from time step 250s to 300s) and the same case run normally from 0 to 300s. Velocities at several distances from the floodgate are compared for both cases, after time step 300s.

For comparing velocities, an imaginary line passing through a point C (10, 1.9, 5) is assumed. The line lies in a plane parallel to the X-Z plane passing through Y=1.9, and makes an angle θ with the partition wall as shown in Figure 3.12. Values of velocities, calculated using both

simulation methods, are sampled for 1000 points along the line. This is done for θ values of 30, 60 and 90 degrees to obtain Figures 3.13, 3.14 and 3.15.

Figures 3.13, 3.14 and 3.15 demonstrate that the difference between the two cases is negligible. Therefore, it is concluded that using mapFields to reduce simulation yields reliable results and can be used for all future OpenFOAM simulations for this analysis.

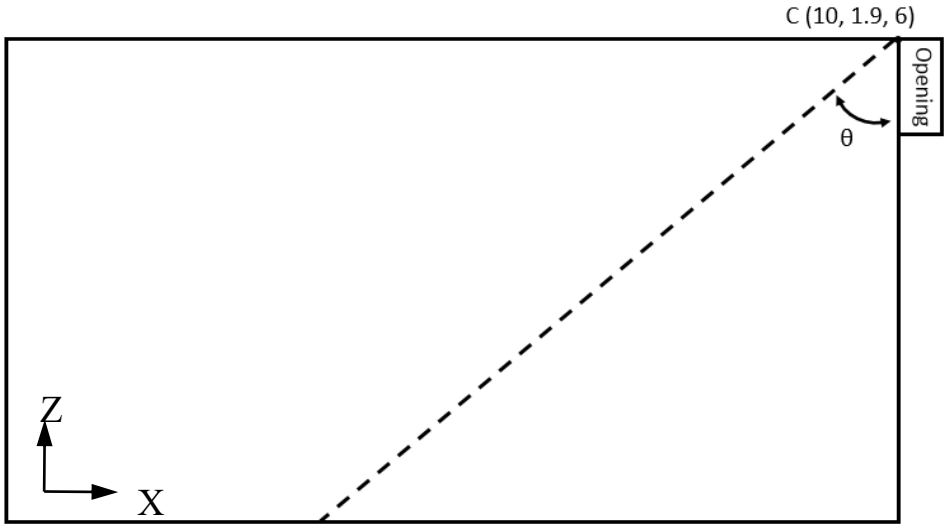

Figure 3. 12: Method for comparison of normal and mapFields method. Picture shows the line along which data for several points are compared for case 1 in Table 1 with and without the use of mapFields.

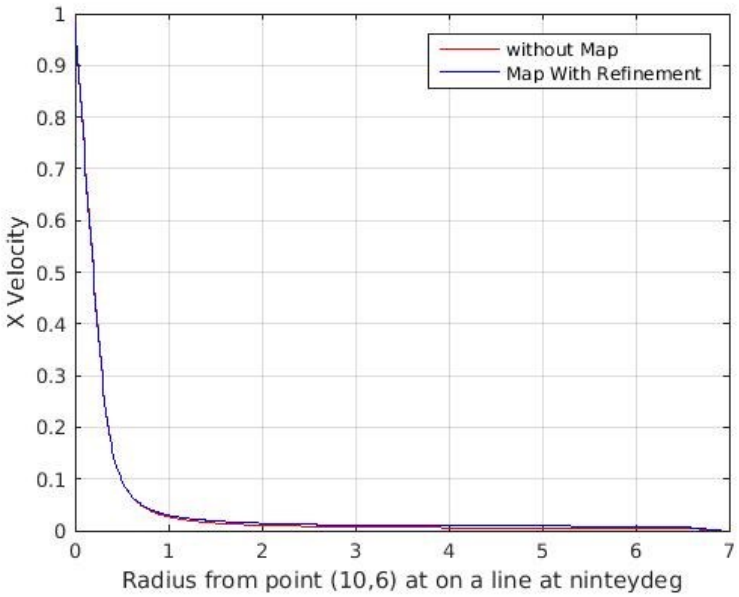

Figure 3. 13: Comparison of velocity with and without the use of mapFields, for θ = 30° in Figure. 3.12

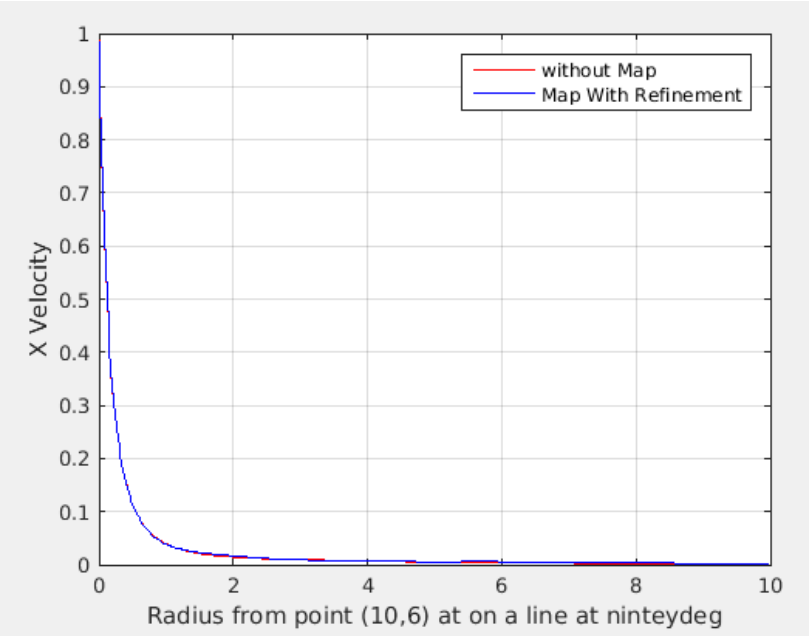

Figure 3. 14: Comparison of velocity with and without the use of mapFields, for θ = 90° in Figure 3.12

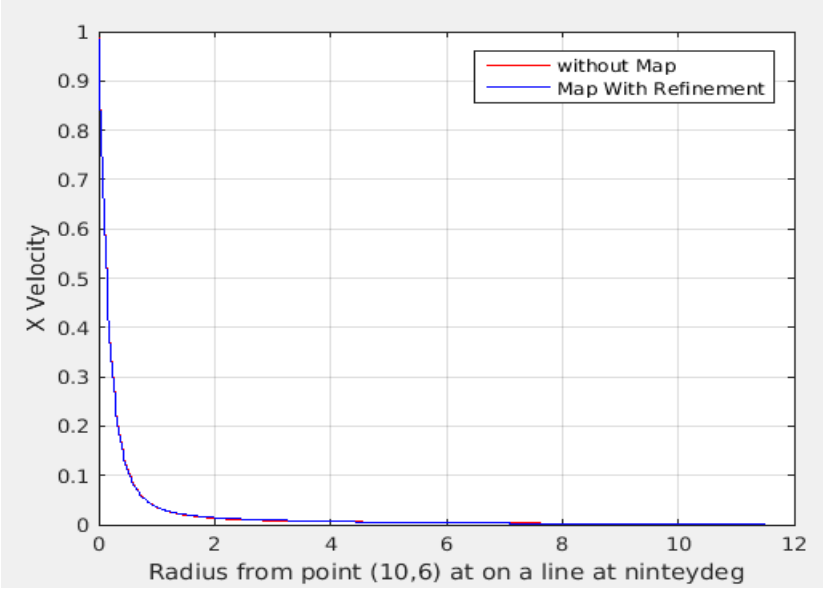

Figure 3. 15: Comparison of velocity with and without the use of mapFields, for θ = 60° in Figure 3.12

The cases in Table 1 are simulated using mapFields for grid convergence in X-Z direction and several velocity contours are compared in Figure 3.16. This figure presents contours of 1%, 2%, 5%, 10%, 15% and 20% of the maximum velocity in each of the case in Table 1. The contours for different cases are very close to each to other especially the ones closer to the floodgate. Contours from case 3 (50000, aqua color) and case 4 (750000, green color) at most places lie between those from case 1 (230000, red) and case 2 (350000, dark blue). Which means that the solution for the 750000 (finest) and 50000 will lie somewhere in between that of the case 1 and 2. Taking a combination of maximum and minimum cell sizes from case 1 and 2 will yield reliable results.

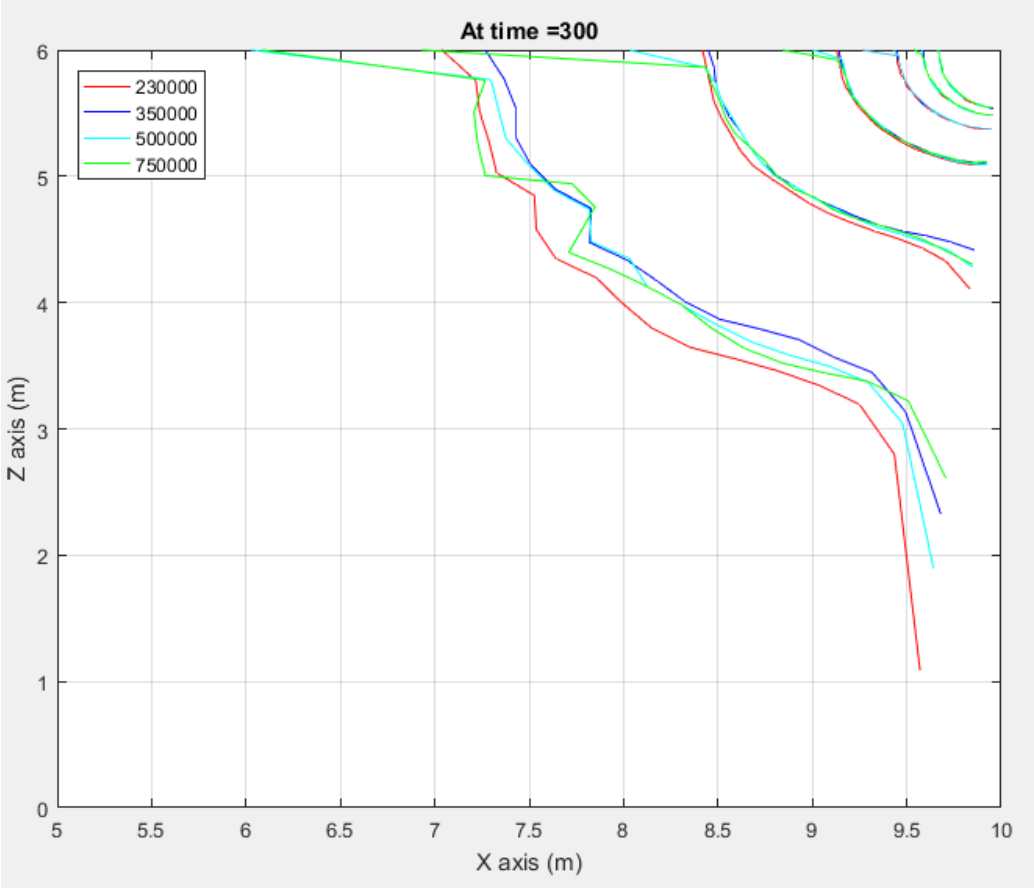

Figure. 3. 16: Velocity contours for 1%, 2%, 5%, 10%, 15% and 20% of the maximum velocity in each case in Table 1, at the end of the simulation (Time = 300s).

For example, the minimum Z cell size for refinement zone b in case 1 is 0.0094 while that in case 2 is 0.0075 (See Table 1). In future simulation meshes, the minimum Z cell size in refinement zone b is a value between 0.0094 and 0.0075. Similarly, for minimum and maximum X and Z cell sizes for other refinement zones in Table 1.

3.8 Grid convergence for cells in Y direction

As mentioned in section 3.3, the grid convergence study is divided into two parts since having refinements in the Y axis distorts the boundary layer (Figure 3.6). Nevertheless, it is necessary to know the number of cells needed near the ice sheet in the Y direction to resolve the boundary layer of the water flow accurately. The accurate resolution of this boundary layer determines the accuracy of the velocities very close to the ice sheet, which are needed

to simulate the motion of the ice piece. In this section, the cell parameters for the Y direction are optimized.

The grid convergence study for Y direction is done with a mesh that has an ice sheet fully covering the tank. The ice sheet is modelled as a wall with no slip boundary condition. Results from the grid convergence study in X and Z directions are imported into this study. The mesh is finer near the ice sheet and progressively becomes coarser. Three mesh cases with different cell parameters (mentioned in Table 4) are prepared. Comparisons of resultant wall shear stresses and boundary layer profile between these cases are done.

The line diagram of the domain is shown in Figure 3.17. Similar to section 3.3.1, a floodgate of 0.5m (with the symmetry plane) is used for this study. The total width of the tank is the same as in section 3.3.1, that is, 5m. Since the entire tank is covered with ice sheet, the "air" portion mesh above the ice sheet (Figure 3.7) has been eliminated.

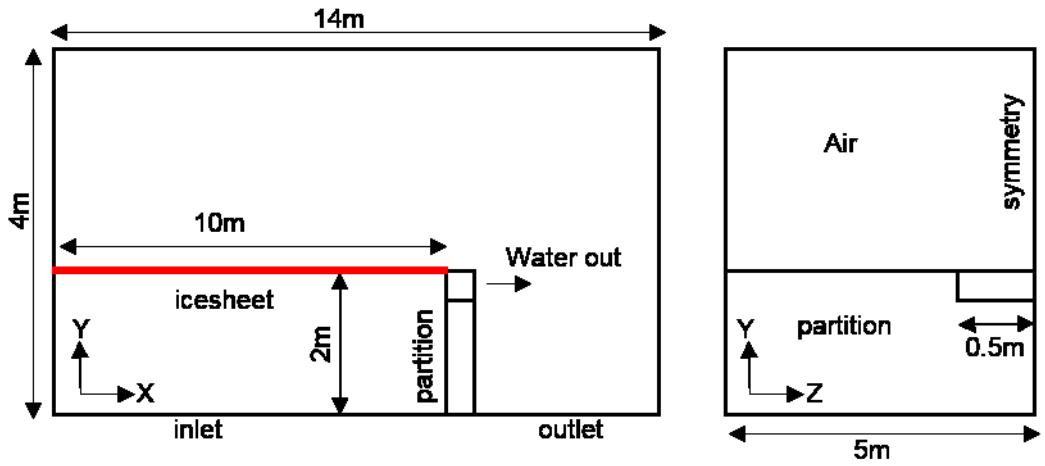

Figure 3. 17: Geometric layout of the mesh used for grid convergence in Y direction

The minimum height of the cell in the Y direction and the total number of cells for each case for three simulation cases are shown in Table 4. The Y grading of the cells is such that cell immediately next to the ice sheet has the smallest cell height. The height of the subsequent cells away from the ice sheet, increase with the cells near the bottom inlet being much larger in comparison. The number of cells in Y direction in a 0.1m band (Figure 3.18) near the ice sheet is also mentioned in Table 4. This parameter indicates the density of cells in Y direction near to the ice sheet.

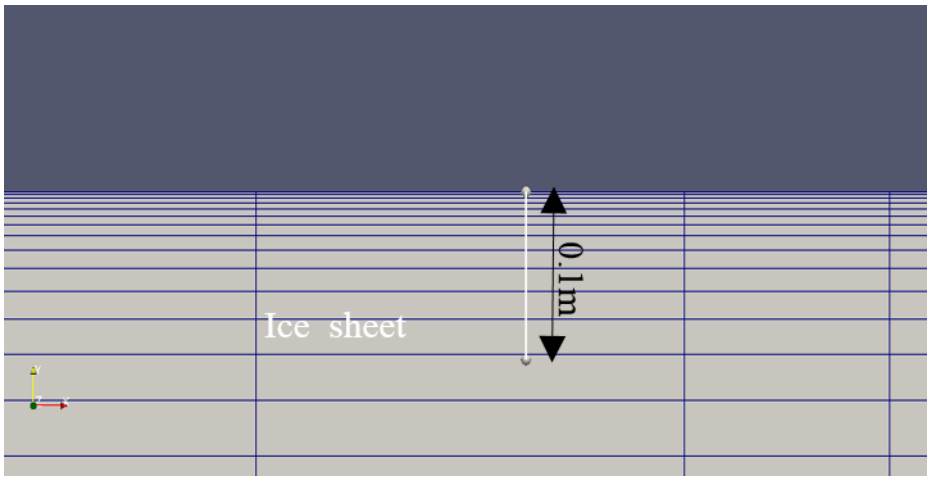

Figure 3. 18: Number of cells in the Y direction contained in a band of 0.1m near the ice sheet

Table 4: Cell parameters for the grid convergence in Y direction

Case	**Total cells in Y direction**	**Minimum cell size in Y direction (Figure 3.18)**	**No. of cells in Y direction in 0.1m band**	**Total number of cells in the mesh**
1	20	0.00123	13	368448
2	25	0.00085	19	495016
3	35	0.00052	32	765931

The addition of ice sheet in the mesh prevents (dampens) formation of any propagating waves (like in Figure 3.11) near the floodgate. As a result, the final time step in these simulations is much lower than that in section 3.3.1.

The simulation is run using mapFields similar to section 3.3.1. The X-Z cell parameters for the coarse mesh is the same as in Table 3. The Y cell parameters for the coarse mesh are the same as in case 1 of Table 4. The boundary conditions are the same as in section 3.3.1. The newly added ice sheet is modelled similar to any other wall in the domain. The coarse mesh is run from time step 0 to 80s. The flow field at time 80s from the coarse mesh is mapped on to the Y grid convergence simulation cases. These simulation cases for Y grid convergence contain cell parameters as a combination results from section 3.3.1 for X-Z cell parameters, and Table 4 for Y grid parameters. These cases further simulated from time step 80s to 100s.

The velocity gradient along the ice sheet is calculated using the wallGradU utility of OpenFOAM. The wallGradU gives components of velocity gradients at a point on the wall in three different directions. The resultant velocity gradient is given by the square root of the sum of squares of the components. The velocity gradient at five random points on the ice sheet for each of the cases in Table 4 is compared in Table 5.

Table 5: Comparison of velocity gradient wallGradU values for cases in Table 4

Point location	**Case 1**	**Case 2**	**Case 3**
(7, 2, 2.5)	0.4188	0.4206	0.4215
(5, 2, 4)	0.321	0.322	0.323
(4, 2, 1.5)	0.202	0.2025	0.2029
(8, 2, 4.5)	1.5168	1.553	1.559
(6, 2, 2)	0.3051	0.3062	0.3068

The difference between wall velocity gradient values for the coarsest case (Table 5, Case 1) in Y and the finest case (Table 5, Case 3) in Y, is insignificant and can be considered negligible. Thus, making the mesh finer than Case 1 near the ice sheet does not improve the accuracy considerably but increases the cell count considerably (Table 4). As a result, Y directional cell parameters from case 1 in Table 3 are used in all future simulations.

4 SIMULATION OF PARTIAL ICE SHEET CASES

While extracting ice pieces one by one using a floodgate, the length and width of the ice sheet decreases. Starting from the tank being fully covered by ice sheet until there is no more ice sheet. In order to have an accurate assessment of the reach time from different starting locations, it is necessary to know the effect of the length of the ice sheet on the reach time. It is also necessary to know whether this effect is insignificant enough to be avoided so that only a single OpenFOAM simulation case is needed to simulate the ice piece motion.

In this analysis, only the change in length of the ice sheet is taken under consideration. The change in width is ignored since the sequence of the cutting ice is unknown. However, despite the sequence being unknown, it is certain that the ice sheet will go from fully covering the tank to completely removed from the tank. Therefore, the ice sheet length will decrease independent of the cutting pattern and sequence of the ice pieces.

4.1 Modelling of ice sheet length variation

Using the optimal mesh parameters obtained from section 3 and the mesh layout mentioned in Figure 4.1, meshes for five different ice sheet lengths are considered for simulation. The length of the ice sheet for these five cases is expressed as the ratio of the ice sheet length to the length of the tank. This ratio is called ice sheet length ratio. The ice sheet length ratio for these five cases is 1 (tank fully covered), 0.75 (75% of tank covered), 0.5 (half covered), 0.25 (25% of tank covered) and 0 (no ice sheet).

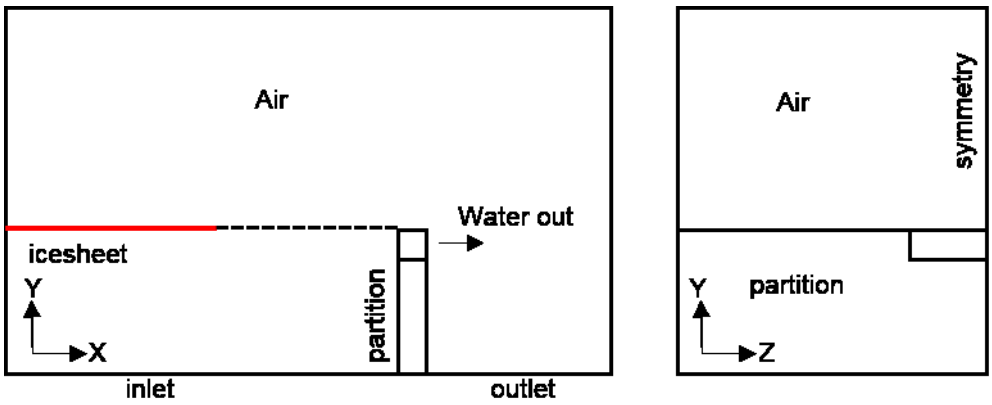

Figure 4. 1: Mesh Layout used to simulate partial ice sheet cases. The length of the ice sheet (red line) is variable depending on the partial ice sheet case.

4.2 Process for running the simulations and exporting results

For each case, mapFields is used similar to section 3 to reduce the total simulation time. A coarse mesh is made for each case such that the number of cell refinements (from refineMesh utility) are less compared to the actual mesh. The assumptions and boundary conditions for this coarse mesh are unchanged from section 3.4 and 3.5 respectively. The simulation for each ice sheet case is run using the following process:

1. Simulation on this coarse mesh is run from initial time step 0 to final time step 80 seconds.
2. Using mapFields, the flow field from this coarse simulation at time step 80 is mapped onto the actual mesh.

3. The simulation on the actual mesh is then run from initial time step 80 to final time step 100.
4. The simulation is assumed to reach a quasi-steady state at time step 100. However, in order to account for miniscule differences between different time steps, a time average of the velocity flow field is taken from time step 90 to time step 100.
5. For simulation of ice piece motion, it is necessary to know the velocity field very close to the ice sheet to accurately calculate the induced drag force on the ice piece. The velocity profiles are taken in layers in XZ plane. Ten such layers are taken from Y = 1.98 to Y = 1.998. The distance between two adjacent layers is 0.002m.
6. An average of velocity values between these layers at the same XZ location is taken to get an average velocity flow field very close to the ice sheet. This average flow field is exported to a text file for ice piece motion simulation. The text file contains the X,Y and Z cordinates of the cell points and the respective velocity components.
7. Steps 1 to 6 are repeated again for other partial ice sheet cases.

The resultant flow fields thus obtained from the simulation of all partial ice sheet cases are used to simulate the ice piece motion.

5 SIMULATION OF ICE PIECE MOTION

The simulation of ice motion and water flow is decoupled due to several factors discussed in section 2. As the ice sheet is cut and the cut pieces are slowly taken out of the tank, the length of the ice sheet reduces. In order to study the effect of this reduction of length on the motion of ice piece, water flow simulation for several ice sheet length cases is done in section 4. The time averaged velocity values from each case is exported to a file such that it can be read by the MATLAB script. The MATLAB script calculates the instantaneous forces acting on the ice piece at every time step, based on its instantaneous position in the tank, and thus calculates its new position, velocity and acceleration from the force balance equation.

When a rectangular ice piece is newly cut from the ice sheet under the influence of the floodgate flow, the ice piece will try to overcome the ice-ice friction between its side wall and the ice sheet, come of out the ice sheet cut groove and travel towards the floodgate. Thus, the motion of the ice piece can be generalized into three phases:

1. Freshly cut ice piece trying to overcome ice-ice friction
2. Ice piece out of the groove and moving freely towards the floodgate
3. Ice piece reaching the floodgate

Stockstill et. al. (Stockstill et. al., 2009) presented a model that uses discrete element mechanics to model the forces on particles in a river flow. Here, the inter-particle interaction, buoyancy forces and fluid drag forces acting on the floating bodies are taken into account. The force balance formula used for the forces acting on a single floating body is:

$$\boldsymbol{m_i \frac{dw_i}{dt} = F_i^f + F_i^p + m_i g} \qquad (5.1)$$

Where,

m_i = Mass of the particle i

F_i^f = Fluid drag force

F_i^p = Fluid pressure force

$\frac{dw_i}{dt}$ = Acceleration of the particle i

g = Acceleration due to gravity

In this case, the ice piece is synonymous to a floating body. Since the Y velocity of the flow near the ice sheet is assumed to be negligible compared to the XZ velocity (see section 3.3.2), all the forces in Y direction are neglected. Effectively, the buoyancy and gravitational forces on the ice piece in the equation 5.1 are neglected. Also, since only a single ice piece is simulated in this study, the interaction forces between multiple ice pieces are also neglected. The term $F_i^f + F_i^p$ in equation 5.1 represent the total resultant drag force acting on the ice piece. However, the drag force acting on a body due to a fluid flow can be expressed as a function of the fluid velocity using the equation:

$$\boldsymbol{F_D = \frac{1}{2} C_D \rho A v^2} \qquad (5.2)$$

Where,

F_D = Drag force acting on the body

C_D = Drag co-efficient

ρ = Density of the fluid

A = Reference (effective) area of the body

v = Relative velocity of the fluid w.r.t. the body

Using equation 5.1 and 5.2, the following forces need to be calculated for every time step:

1. Drag force using the velocity flow field obtained from section 4
2. Ice-Ice Friction force when the ice piece is cut from the ice sheet (if the ice is inside the cut groove).

5.1 Assumptions for Implementation in MATLAB

The forces listed above are calculated on a single ice piece, based on the water flow simulated in OpenFOAM, from which motion parameters of the ice piece such as acceleration, velocity and location for each time step are calculated, using the force balance equation.

To reduce the complexity of the simulation and to make it practical, the following assumptions are considered:

1. **Ice piece is a rigid body:** The ice piece is assumed to be a rigid body such that any deformation and cracking of ice, after it is cut from the ice sheet, is neglected. Under this assumption, motion of the ice piece is calculated by just tracking the position of the center of the mass of the ice piece for each time step.
2. **Dimensions of the ice piece:** The ice piece cut from the ice sheet is assumed to be a square of length, width and thickness as 1m, 1m and 0.02m respectively.
   Hasan and Louhi-Kultanen (Hasan and Louhi-Kultanen, 2015) have presented a model for ice growth for air cooled sodium sulfate solutions with varying concentrations of 1, 2 and 3% by weight. In this, the maximum calculated thickness of the ice layer, which is formed at 3% concentration for a temperature difference (between water and air) of 3°C, is approximately 0.02m. Hence, the thickness of the ice sheet is assumed as 0.02m
3. **Rotation of the ice piece is ignored:** The aim of the simulation is to predict the time taken for an ice piece from its cut position to the floodgate, for different floodgate width. This is assumed to be adequately done by calculating just the translation of the ice piece.
4. **Collision with walls:** When the ice piece hits the partition wall or the walls of the tank, it will eventually drag and roll itself towards the floodgate. This is why the friction between the walls and ice piece is neglected and the ice piece is assumed to slide frictionlessly over the wall.
5. **Movement of the ice piece in Y direction is ignored:** During the grid convergence test for X and Z direction (section 3.3.1), the Y velocities have been found to be negligible compared to the flow velocity in the X and Z direction. The flow can be laminar and only in X and Z direction near the ice sheet. Thus, it is assumed that the influence of Y velocities of water flow can be neglected completely and only X and Z motion of the ice piece are taken into account.
6. **Density of the ice:** Density of pure ice at various temperatures below freezing are documented by (Hobbs, 1974). Here, an average value of density from 0 °C to -5 °C since

the surrounding air temperatures are uncertain. The density of ice based on this calculated average is assumed as 917 kg/m^3.

7. **No wedging action between ice and ice sheet:** Since the rotation of the ice piece is ignored, it is assumed that the rectangular ice piece does not wedge itself diagonally between the ice sheet cut groove. It is assumed that the cutting is done in such a way that the wedging process is avoided.
8. **Water does not refreeze:** During the ice piece motion from its initial cut location towards the floodgate, the water on which it is transported is assumed to not refreeze to form a new ice sheet. It is assumed that the rate of freezing (for purification) is low enough for this to not happen.
9. **Added mass of the thin ice piece:** Added mass or virtual mass is due to an accelerating or decelerating body deflecting the fluid through which it moves. It stems from the fact that the fluid and the body cannot occupy the same space. For a rectangular flat plate, the added mass for movement in directions along the plane of the plate is zero (Clauss, 2014). In this problem, the length and width of the ice piece (1m) are much larger compared to its thickness (0.02m). Therefore, the added mass of the ice piece is neglected.
10. **Kinetic ice-ice coefficient of friction:** The kinetic coefficient of friction for the friction between two ice pieces is interpolated from (Schulson and Duval, 2009) which gives kinetic friction coefficient between two pure ice pieces at temperatures -3, -10, -30 and -40° C for sliding speeds ranging from 10^{-7} to 10^{-1} m/s. The coefficient value in this case is interpolated from the temperature chart for -3° C between sliding velocities 10^{-2} and 10^{-1} m/s. The kinetic friction coefficient for ice-ice friction is assumed to be 0.05 from this interpolation.
11. **Drag Coefficient of the Ice piece:** Equation 5.2 gives the total drag force acting on the ice piece as a function of the fluid velocity. For this, the drag coefficient is assumed to be 1.2 (BS 5400-2:2006 Steel, concrete and composite bridges).

5.2 Calculation of Drag Force

Using equation 5.2, the drag force acting on a body as a function of the relative velocity between the body and the fluid can be calculated. Instead of calculating the resultant drag force, it is split into its directional components such that each component of the drag flow is a function of the respective directional flow velocity component.

5.2.1 Calculating average flow velocity at the center of the mass

As mentioned in section 5.1, the ice piece is assumed to be rigid and thus the velocity of the ice piece can be calculated at its center of mass. However, the flow velocity of the water close to the ice piece is different at different points of the mesh depending on the location of the ice piece. A weighted average velocity of the water near the ice piece acting at the center of mass of the ice piece is found. By subtracting the velocity of the ice piece from the average water flow velocity at center of mass, the relative velocity between the ice piece and the water near to the ice piece, is obtained.

As the ice piece moves inside the tank, it will engulf multiple cell points (of the OpenFOAM mesh). To find the average X and Z velocity values at the center of mass (CM) of the ice piece, weighted averages of the velocity values at the cell points contained within the ice piece are calculated based on the distance of the cell points from the CM. The closer the cell point is to the ice piece, the more weightage it should have on the average velocity. Therefore, the weight function used is the inverse of the square of the distance between CM and the cell point.

In Figure 5.1, d_i represents the distance of the cell point from the center of mass of the ice piece and v_i represents the value of the velocity at that point. The weighted average velocity at the center of mass (v_{cm}) is calculated using equation 5.3.

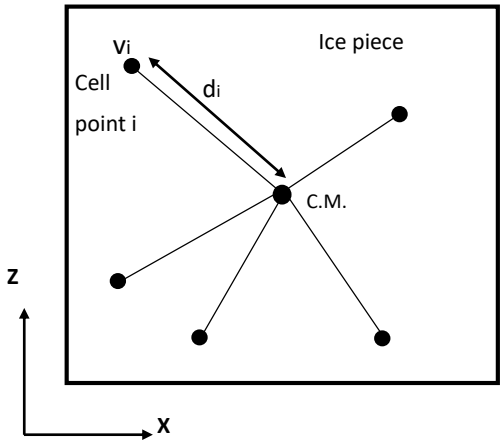

Figure 5. 1: Calculation of average shear stress value at the center of mass (CM)

$$\boldsymbol{v_{cm}} = \frac{\sum \boldsymbol{v_i} \frac{\mathbf{1}}{\boldsymbol{d_i^2}}}{\sum \frac{\mathbf{1}}{\boldsymbol{d_i^2}}} \qquad (5.3)$$

As per equation 5.3, the X and Z components of velocity at the center of mass (v_{cm}^x and v_{cm}^x) are given by equations 5.4 and 5.5 respectively.

$$\boldsymbol{v_{cm}^x} = \frac{\sum \boldsymbol{v_{xi}} \frac{\mathbf{1}}{\boldsymbol{d_i^2}}}{\sum \frac{\mathbf{1}}{\boldsymbol{d_i^2}}} \qquad (5.4)$$

$$\boldsymbol{v_{cm}^z} = \frac{\sum \boldsymbol{v_{zi}} \frac{\mathbf{1}}{\boldsymbol{d_i^2}}}{\sum \frac{\mathbf{1}}{\boldsymbol{d_i^2}}} \qquad (5.5)$$

The relative velocity between the ice piece and the water flow near to the ice piece, in the X and Z direction, at any instant is obtained by,

$$\boldsymbol{v_{rel}^x} = \boldsymbol{v_{cm}^x} - \boldsymbol{v_{ice}^x} \qquad (5.6)$$

$$\boldsymbol{v_{rel}^z} = \boldsymbol{v_{cm}^z} - \boldsymbol{v_{ice}^z} \qquad (5.7)$$

Where, v_{ice}^x and v_{ice}^z are the X and Z components of the ice piece velocity.

Using the relative velocities from equation 5.6 and 5.7, the X and Z components of the drag force acting on the ice piece is obtained by,

$$F_D^x = \frac{1}{2} C_D \rho_{water} A_{side} (v_{rel}^x)^2 \qquad (5.8)$$

$$F_D^z = \frac{1}{2} C_D \rho_{water} A_{side} (v_{rel}^z)^2 \qquad (5.9)$$

A_{side} is the area of side of the ice piece. Since the ice piece is assumed to be a square of 1m length, the side area in this case is equal to the thickness of the ice piece. Figure 5.2 shows the components of the drag forces and the resultant drag force acting at the center of mass of a newly cut ice piece.

5.3 Calculating Ice-Ice Friction Force

For a freshly cut ice piece, based on the location and the Z velocity of the water, the ice piece has to overcome the ice-ice friction force in order to get out of the groove. The nature of the water flow next to the ice piece is such that the Z velocity (if not negligible or zero) is always pointing towards the floodgate. This is because the floodgate behaves like a sink in this situation. Due to this, only one side ice piece (towards which the Z velocity points at) will face ice-ice friction. The resultant drag force and its component is show in Figure 5.2.

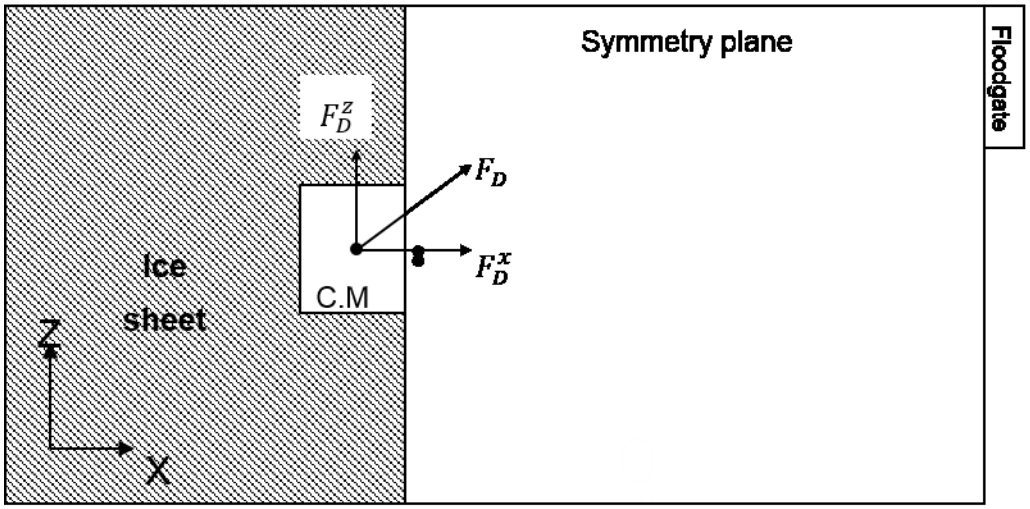

Figure 5. 2: Diagram showing the component and resultant drag force on a newly cut ice piece

Once the ice piece is cut from the ice sheet, it is completely detached from any sides of the uncut ice sheet by default. However, due to the drag force in the Z directions, the ice piece will move such that it will be in contact with the ice sheet on one side. The normal force exerted by the ice piece on the ice sheet side is Z component of the drag force as shown in Figure 5.3. Using this normal force, the ice-ice friction force (F_f) is calculated by:

$$F_f = \mu_d F_d^z \qquad (5.10)$$

Where, μ_d = Dynamic co-efficient of friction for ice and ice.

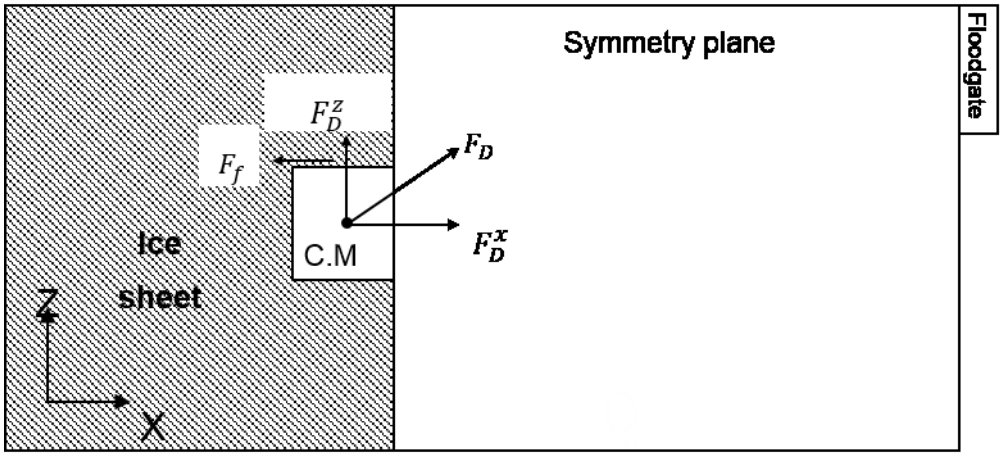

Figure 5. 3: Combination of drag force along with ice - ice friction force on a newly cut ice piece

For the ice piece to move forward, the drag force in the X direction (F_d^x) should be greater than the friction force (F_f). If $F_d^x > F_f$, the ice piece will slowly accelerate only in X direction till the point that it is outside the ice sheet cut groove. While the ice piece is inside the groove, the ice sheet will not allow it to move in Z direction. Thus, Z velocity, acceleration and displacement are simply taken as zero (or not calculated) during this phase. Once outside the cut groove, away from any contact with the uncut ice sheet, the ice-ice friction forces will be zero, and the ice piece will accelerate towards the floodgate due to drag forces in both X and Z directions.

5.4 Calculating acceleration, velocity and position of the ice piece

Applying force balance equation on the ice piece, several motion parameters for the ice piece such as its acceleration, velocity and position at a particular time step are calculated. The force balance equation for the ice piece at any instant is given by,

$$\boldsymbol{m_{ice}\, a_{ice}^{k} = F_D^k - F_f^k} \qquad (5.11)$$

Where,

m_{ice} = Mass of the ice piece,

a_{ice} = Acceleration of the ice piece

The superscript k represents the current time step. As the time step marches forward from zero, the location of the ice piece in the tank would be different if the ice piece is in motion. This means that the forces and thus, acceleration, acting on the ice piece is subject to time. Due to this reason, the force balance equation is solved to get acceleration at every time step.

Based on the acceleration for the current time step, the velocity for the next time step can be obtained by using the definition of acceleration in equations 5.12 and 5.13.

$$\frac{\partial V_{ice}}{\partial t} = a_{ice} \quad OR \quad \lim_{\Delta t \to 0} \frac{\Delta V_{ice}}{\Delta t} = a_{ice} \qquad (5.12)$$

Here, V_{ice} is the velocity of the ice piece. For an infinitesimally small and positive time step,

$$\frac{V_{ice}^{k+\Delta t} - V_{ice}^{k}}{\Delta t} = a_{ice}^{k} \quad OR \quad V_{ice}^{k+\Delta t} = \left(a_{ice}^{k}.\Delta t\right) + V_{ice}^{k} \qquad (5.13)$$

A newly cut ice piece will be at rest at the beginning for $k = 0$. Thus, $V_{ice}^{0} = 0$. This is the initial condition for the ice piece simulation. Using this initial condition and the equation 5.13, the velocity of the ice piece for the next time step is calculated. This process is repeated for each time step till the ice piece reaches the floodgate. The drag forces are calculated as components in X and Z, and using these values in the force balance equation (equation 5.11) give the respective components of acceleration. These components of acceleration used in equation 5.13 give components of ice piece velocity for each time step.

Similar to the above process, using definition of velocity,

$$\frac{\partial S_{ice}}{\partial t} = V_{ice} \quad OR \quad \lim_{\Delta t \to 0} \frac{\Delta S_{ice}}{\Delta t} = V_{ice} \tag{5.14}$$

Here, S_{ice} is the position of the center of mass of the ice piece in the tank. For a infinitesimally small and positive time step,

$$\frac{S_{ice}^{k+\Delta t} - S_{ice}^{k}}{\Delta t} = V_{ice}^{k} \quad OR \quad S_{ice}^{k+\Delta t} = \left(V_{ice}^{k} . \Delta t\right) + S_{ice}^{k} \tag{5.15}$$

For a newly cut ice piece, the initial position (cut location) of the center of mass of the ice piece is defined in the beginning as initial conditions for the position of the ice piece. Using equation 5.15 and the velocity at that time step, the position of the center of mass of the ice piece for the next time step ($S_{ice}^{k+\Delta t}$) is obtained. Since the ice piece is assumed to be rigid i.e. deformation of the ice piece is neglected, the position of the center of mass gives the location of the ice piece for the next time step. This is repeated for all subsequent time steps until the ice piece reach the floodgate.

Figure 5.4 shows the flowchart for the process used to solve the ice piece simulation in MATLAB.

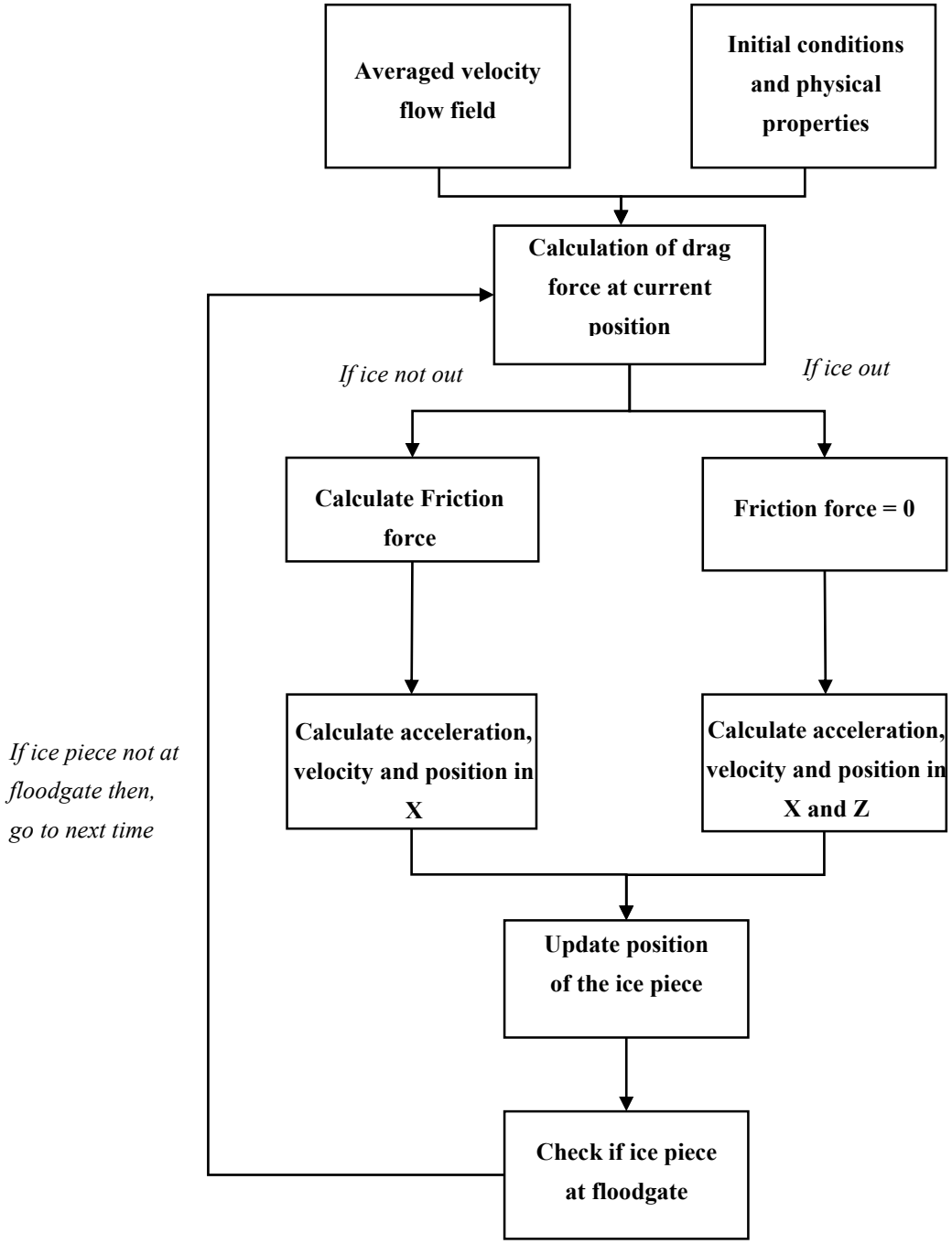

Figure 5. 4: Flowchart for simulation of ice piece motion in MATLAB

1. The files containing the time averaged (90s to 100s) velocity values are imported into MATLAB.
2. Necessary constant parameters such as initial position, density of ice, density and viscosity of water, dimension of the ice piece and dynamic coefficient of friction are defined. These properties and their respective values used for the simulation are mentioned in Table 6.
3. For the initial position (or cut position) of the ice piece, the drag forces for the ice piece is calculated based on the method in Section 5.2.

4. An *iceout* flag tracks whether the ice is inside the cut groove or outside. Check if the *iceout* flag is 1 or 0:
   A. IF 0 THEN ICE PIECE IS INSIDE THE CUT GROOVE:
   a) Ice-ice friction force is calculated based on method mentioned in section 5.3. The Z component of the acceleration, velocity and displacement is set to zero (or not calculated) since the ice piece cannot move in the Z direction due to being stuck in the cut groove.
   b) Based on the method in section 5.4, the X velocity for the next time step is calculated.
   c) The position of the ice piece in the X direction for the next time step is calculated based on the obtained velocity.
   d) Check to see if the back edge of the ice piece for the next time step has come out of the cut groove (Figure 5.5). If so then, the *iceout* flag is set to 1. This means that the ice is out of the cut groove. If the back edge has not come out of the cut groove, *iceout* flag is kept as 0.
   B. IF 1 THEN ICE PIECE IS OUTSIDE THE GROOVE:
   a) Ice-ice friction force is set to zero since the ice piece is no longer in contact with the ice sheet.
   b) Based on the method in section 5.2, X and Z components drag forces are calculated.
   c) Based on the method in section 5.4, the X and Z components of velocity of the ice piece are calculated for the next time step.
   d) Based on the velocity in X and Z directions, the position of the ice piece is calculated for the next time step.
   e) Check if the front edge of the ice piece is in contact with the wall. If the front edge of the ice piece is in contact with the wall, the X acceleration and velocity is set as zero and henceforth only the Z acceleration and velocities are calculated. This is in line with the assumption that the ice will somehow roll and slide its way towards the floodgate.
5. Check if the front edge of the ice piece has reached the floodgate (Figure 5.5). If so, then simulation is ended. If not, then the time step is increased by Δt and the process is repeated from step 3.

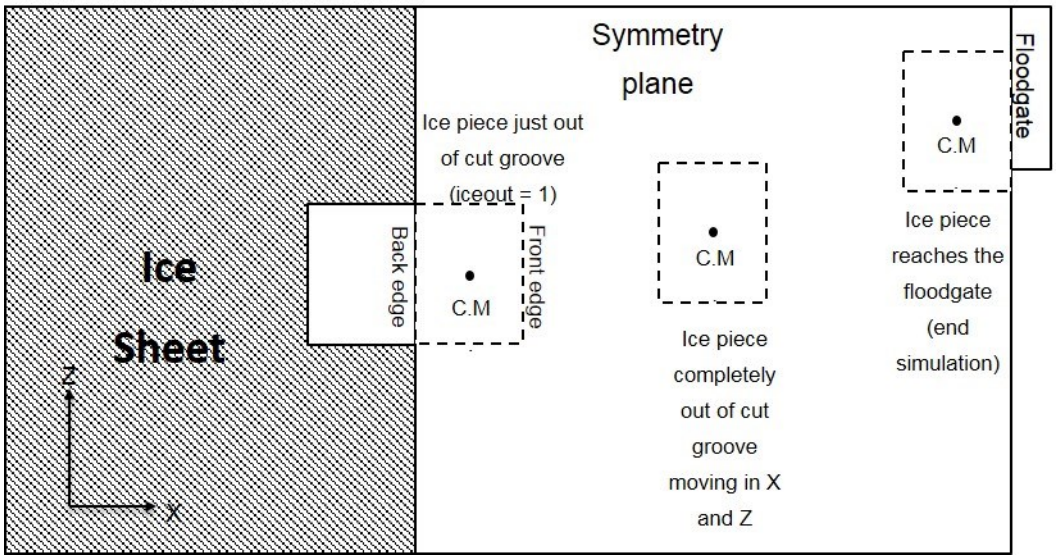

Figure 5. 5: Motion of the ice piece and different stages of the motion. Also describing the events happening in the MATLAB code.

Table 6: Initial parameters and conditions for MATLAB simulation

Initial Parameters	**Initial Value**
Length of ice piece in X direction (X_{len})	1 m
Width of ice piece in Z direction (Z_{len})	1 m
Thickness of ice piece (t)	0.02 m
Density of Ice (ρ_{ice})	917 kg/m^3
Density of water (ρ_{water})	1000 kg/m^3
Thickness of ice submerged in water	$\frac{\rho_{ice}}{\rho_{water}} t =$ 0.0183 m
Area of bottom face of the ice	$A_{bottom} = X_{len}.Z_{len} = 1m^2$
Area of side face of the ice	$A_{bottom} = X_{len}.t = 0.02m^2$
Mass of ice piece	$m_{ice} = A_{bottom}.t.\rho_{ice} = 18.34kg$
Initial acceleration and velocity in X and Z direction	0 m/s
Drag Coefficient	$C_D = 1.2$
Co-efficient of dynamic friction for ice-ice	$\mu_d = 0.05$

6 RESULTS FROM ICE PIECE MOTION SIMULATION

The aims of the thesis (as mentioned in section 1.2) are to simulate the motion of an ice piece due to the opening of the floodgate, study the feasibility of the method in extracting the ice pieces, study the effect of change in ice sheet length on the reach time and study the reach time as a function of the distance from the floodgate. Section 5 goes in depth into the simulation of the ice piece motion while the other three aims are discussed about below.

6.1 Comparisons between partial ice sheet cases

The aim of this comparison is to study effect of the changing ice sheet length on the reach times of the cut ice piece. If the reach time differences between different cases are insignificant, there is no need to simulate multiple ice sheet cases in OpenFOAM to get the flow velocity profile, for future simulations. Rather, a single flow simulation would be sufficient to simulate the ice piece motion.

The ice piece motion simulation is done for several starting locations for all the partial ice sheet cases mentioned in section 4. A grid of 20 x 20 points in the X (from 0.5 to 8.5) and Z (from 0.5 to 4.5) directions is made and the ice piece motion for all the partial length cases is simulated using values from this grid as initial positions of the ice piece. The assumptions and boundary conditions (mentioned in section 5) are the same for all the cases. The only difference is the imported velocity flow field described in section 4. The reach time (t) is recorded for each of the initial positions along with the distance from the center of the floodgate (r).

A color map of the reach time for each case is shown in Figures 6.1, 6.2, 6.3, 6.4 and 6.5.Figure 6.6 shows plots for reach time v/s the distance for all the five cases.

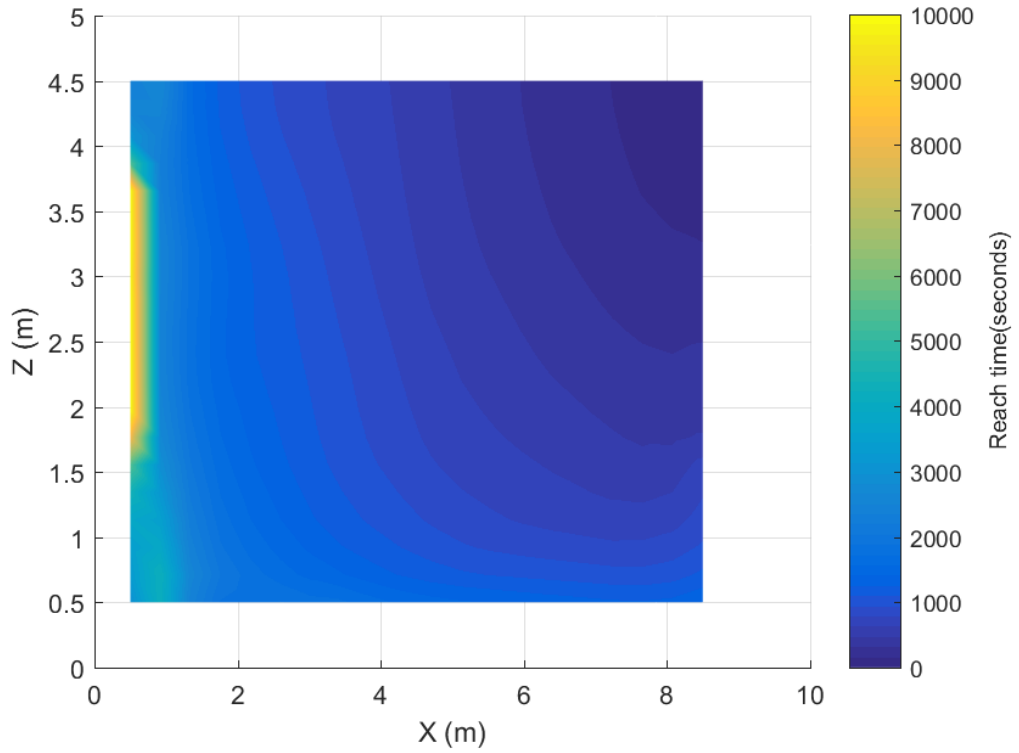

Figure 6. 1: Colormap of reach time with different initial position for 0% ice sheet length case

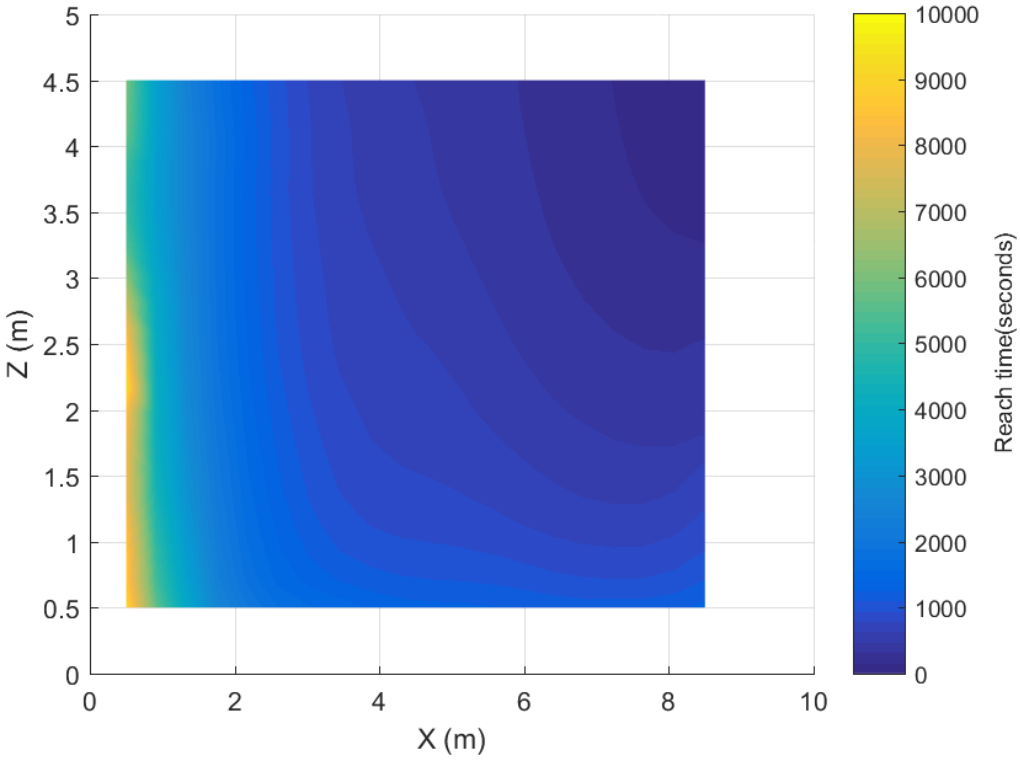

Figure 6. 2: Colormap of reach time with different initial position for 25% ice sheet length case

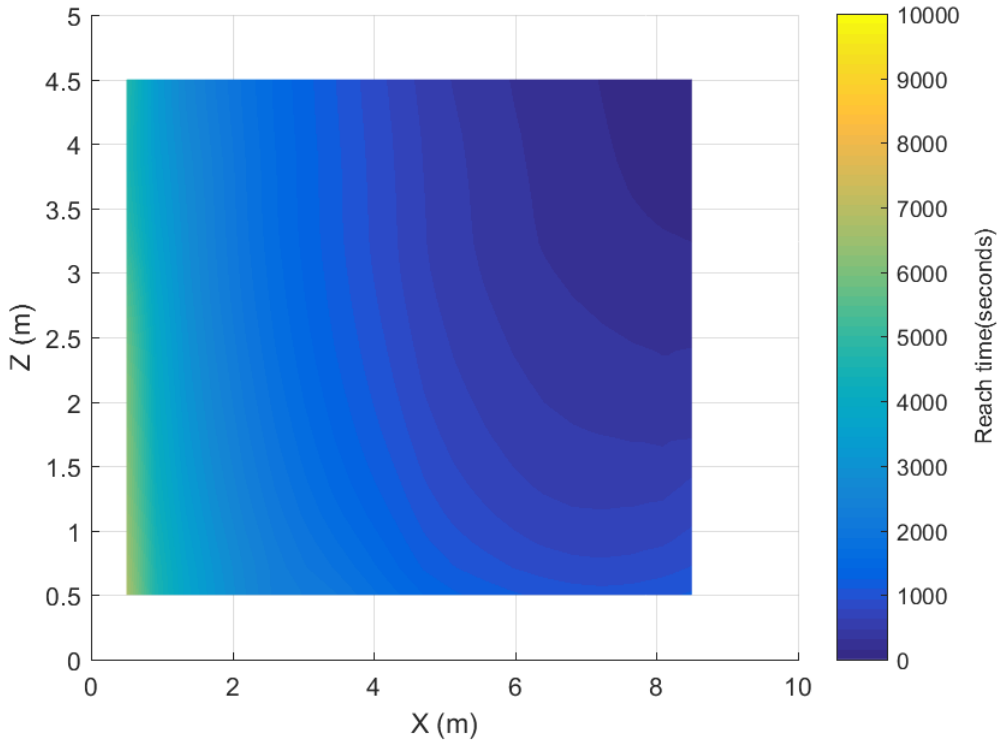

Figure 6. 3: Colormap of reach time with different initial position for 50% ice sheet length case

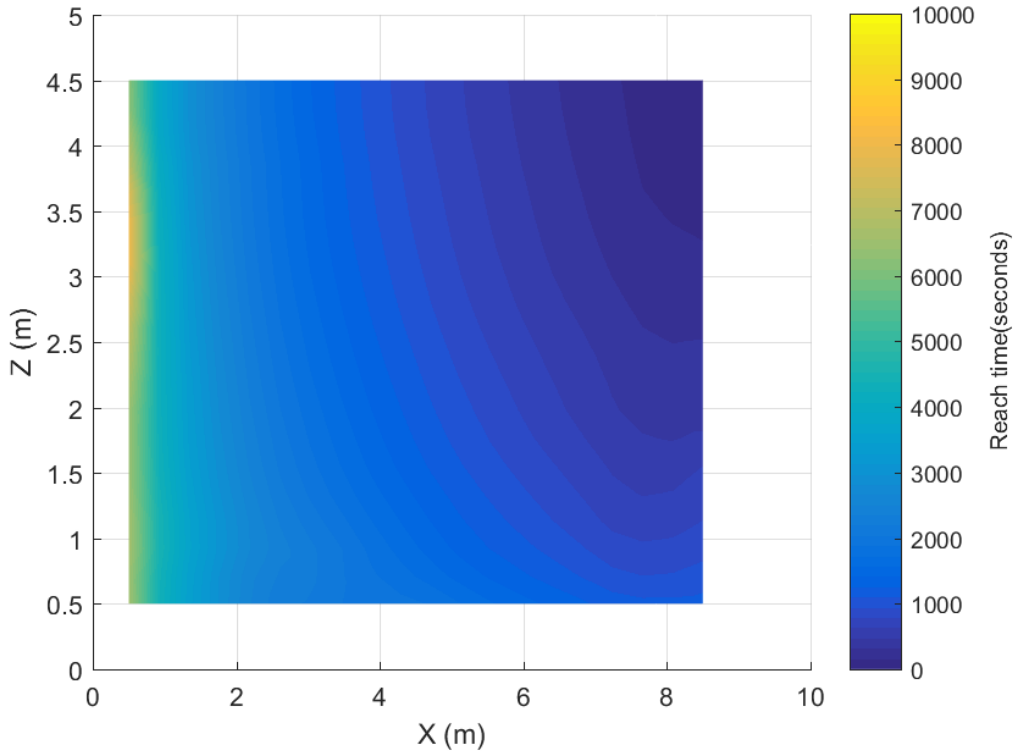

Figure 6. 4: Colormap of reach time with different initial position for 75% ice sheet length case

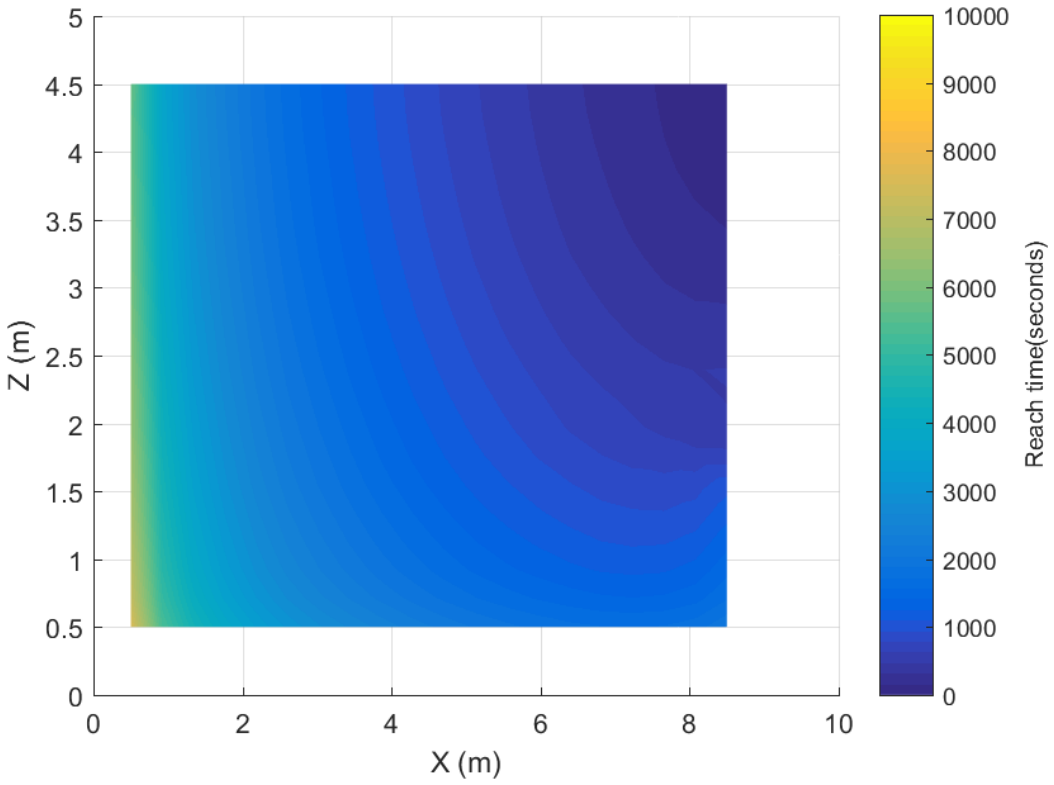

Figure 6. 5: Colormap of reach time with different initial position for 100% ice sheet length case

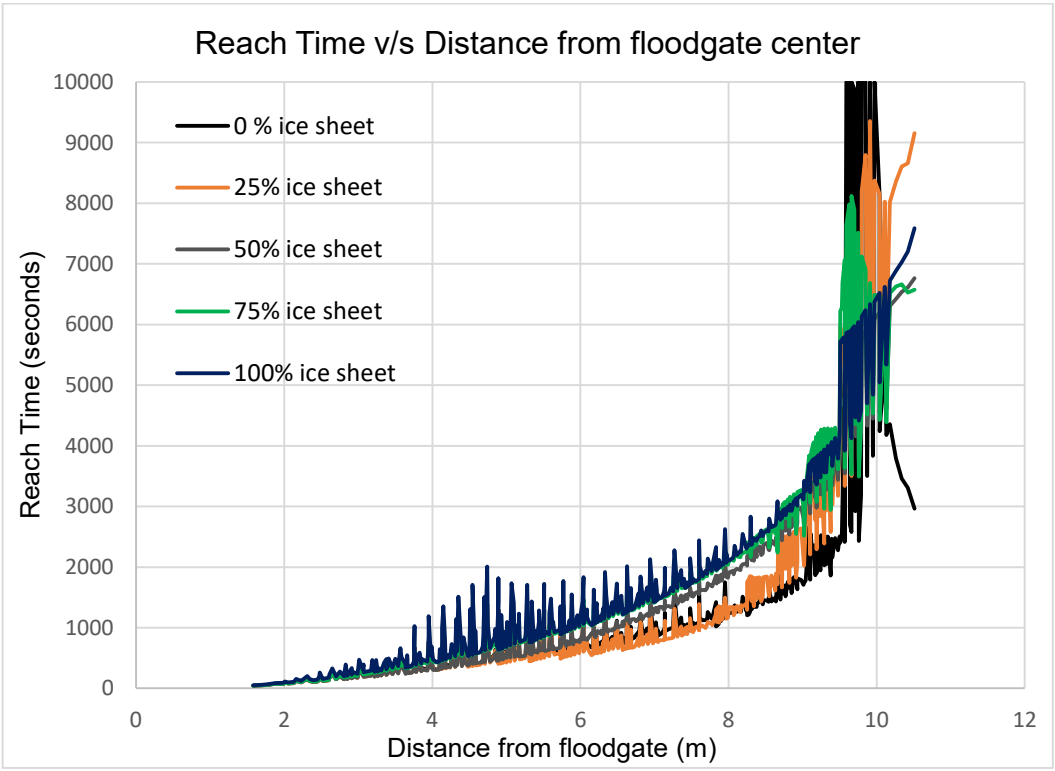

Figure 6. 6: Reach time v/s distance from floodgate plot for different partial ice sheet cases

The colormaps in Figures 6.1 to 6.5 reveal contours of reach time values. In case of 100% ice sheet, the contours lines are circular arcs in nature. However, as the ice sheet length decreases, the contours become more and more distorted. For the 100% ice sheet case, the effect of boundary layer is consistent throughout the entire length of the domain since the ice sheet covers the entire domain. However, as the ice sheet decreases in length, the effect of boundary layer also changes. The boundary layer only affects the length of the domain which is covered by the ice sheet. For fluid layers very close to ice sheet, the velocity is lower for the covered sections compared to the uncovered sections, due to the boundary layer. Therefore, a single simulation case is not used to simulate the ice piece motion. Instead, an interpolation of flow fields from different partial ice cases is used.

By interpolating the flow field using different ice length cases, based on the initial position of the ice piece, an approximation of the water flow field for an ice sheet of that length is generated. For example, if the initial position of the ice piece is at a point (3.5, 2.5), the ice sheet would be of the length 4m. The ice sheet length ratio (refer section 4) is 0.4. The velocity flow field is linearly interpolated using the flow field from the 0.25 and 0.5 cases. Similarly, for an ice sheet length ratio of 0.6, cases 0.5 and 0.75 are used. Since the meshes for all the cases have cell points at the same locations, the entire flow field is interpolated by individually interpolating the velocity values at each cell points.

The ice piece motion is simulated using the same 20 x 20 point grid for the initial position from section 6.2. The color map of the reach time and the plot of reach time v/s distance for this interpolated case is show in Figure 6.7 and Figure 6.8 respectively.

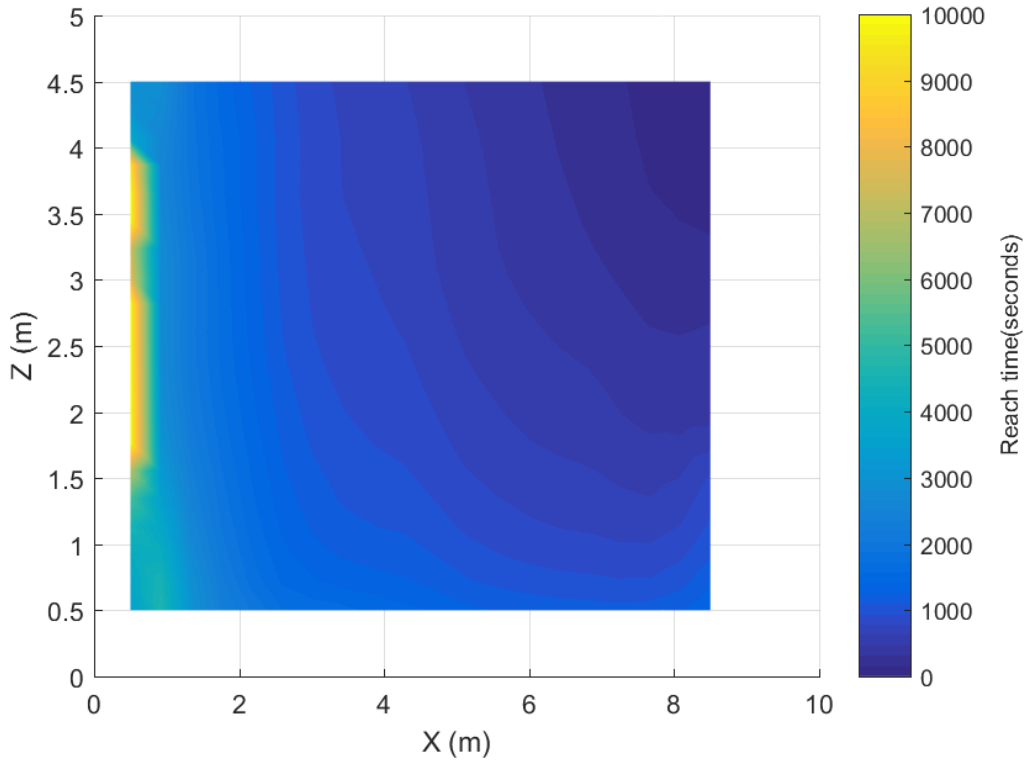

Figure 6. 7: Colormap of reach time based on different initial positions for the interpolated case

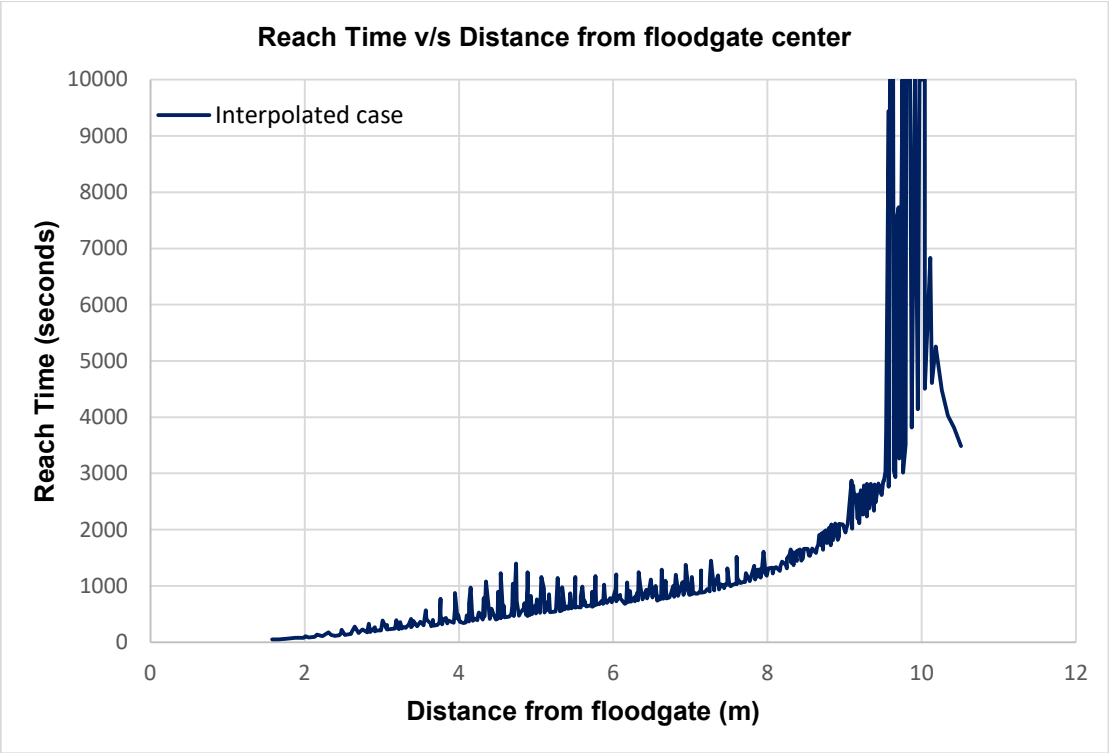

Figure 6. 8: Reach time v/s distance from floodgate plot for different partial ice sheet cases

6.2 Feasibility of the Floodgate method

The behavior of the floodgate in this scenario is similar to a 2D sink (negative source). Equation 6.1 expresses the radial velocity (V) induced by a sink with flow rate Q at any point which is at a distance (R), in polar coordinates.

$$V = -\frac{Q}{2\pi R} \qquad (6.1)$$

The negative sign is represent the fluid flowing out of the domain and the radial velocity vector pointing towards the sink.

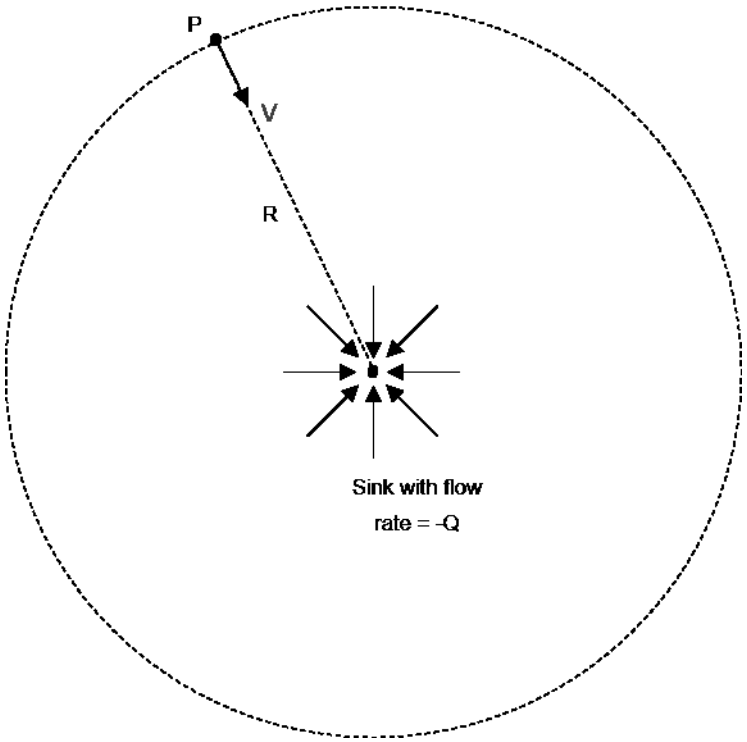

Figure 6. 9: 2D representation of a potential flow in a sink

It is clear from equation 6.1 that the velocity at a point away from the sink is inversely proportional to the distance of the point from the sink. The further the point is away from the sink, the less is the effect of the sink at that point. If the floodgate flow is able to remove the ice piece from the tank starting from the farthest initial position, the flow should be powerful enough for any other initial position.

In case of a rectangular tank where the sink (floodgate) is located at one of the corners of the rectangle, the farthest point from the sink would be the diagonally opposite corner. Therefore, in this analysis, the farthest possible cut position of an ice piece of dimensions 1m x 1m would be at the location (0.5, 0.5). This is the location of the center of mass of the ice piece (Point A in Figure 6.10).

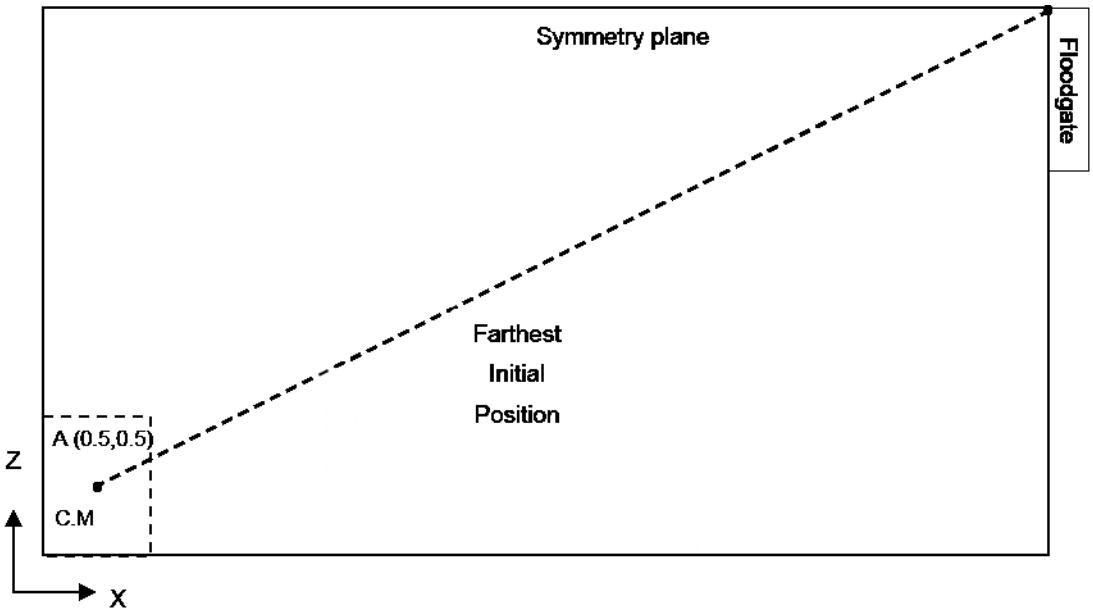

Figure 6. 10: Farthest initial position from the center of the floodgate

Running the interpolated case mentioned in section 6.1, with this initial position, the reach time obtained is 3508 seconds (about 58 minutes).

From Figure 6.7 (which shows the reach time color map in interpolated case) it is clear that ice pieces near the floodgate move quickly and most of the time during extraction will be spent on ice pieces at the opposite end of the tank, from the floodgate. As per the equations 5.8 and 5.9 in Section 5.2, the drag force (and thus the acceleration and displacement) is proportional to the square of the induced relative velocities. Also, from equation 6.1, the velocity induced by the floodgate (sink) is directly proportional to the flow rate of the floodgate. A wider floodgate, with a higher flow rate, will induce higher velocities at the same distance and thus, reduce the reach time. Therefore, this method for extracting ice pieces is deemed feasible since the reach time can be reduced by using a wider floodgate such that the process can be completed in a practical amount of time.

6.3 Analysis of reach time v/s distance from the floodgate

Figures 6.6 and 6.8 show the reach time v/s distance plot for different ice partial ice sheet cases and the interpolated cases respectively. It demonstrates the oscillations present in the reach time values at the same distance.

At a fixed distance from the floodgate, there are multiple possible points within the tank as shown in Figure 6.9. The initial positions of ice piece have different X and Z coordinates even for the same distance. Therefore, for the same distance value, different interpolated partial ice sheet flow fields are used, resulting in variations in reach time for the same distance. This is also corroborated by the reach time contours being distorted in figure 6.7 instead of being circular as per the sink flow theory.

Also, figures 6.6 and 6.8 show that after a certain distance, the reach time values spike to unreasonably high values. It shows that beyond a certain range, the distance ceases to be a good way to measure the reach time. In this case, this distance after which the reach time values spike unreasonably is 9.5m as witnessed from figure 6.7.

Despite the oscillations and the unreasonable spike in the reach time after 9.5m, there is still an observed trend in the reach time v/s distance plot that is valid for most of the domain. In order to analyze the trend of the plot, the reach time values are binned into 16 bins based on the distance. The size of each bin is 0.5. The first bin is from 1.5m to 2m. The next bin is from 2m to 2.5m and so on. The final bin is from 9m to 9.5m. Next, the average reach time and standard deviation for each bin is found. The average (μ) for each bin and the 2σ standard deviation is shown in figure 6.11.

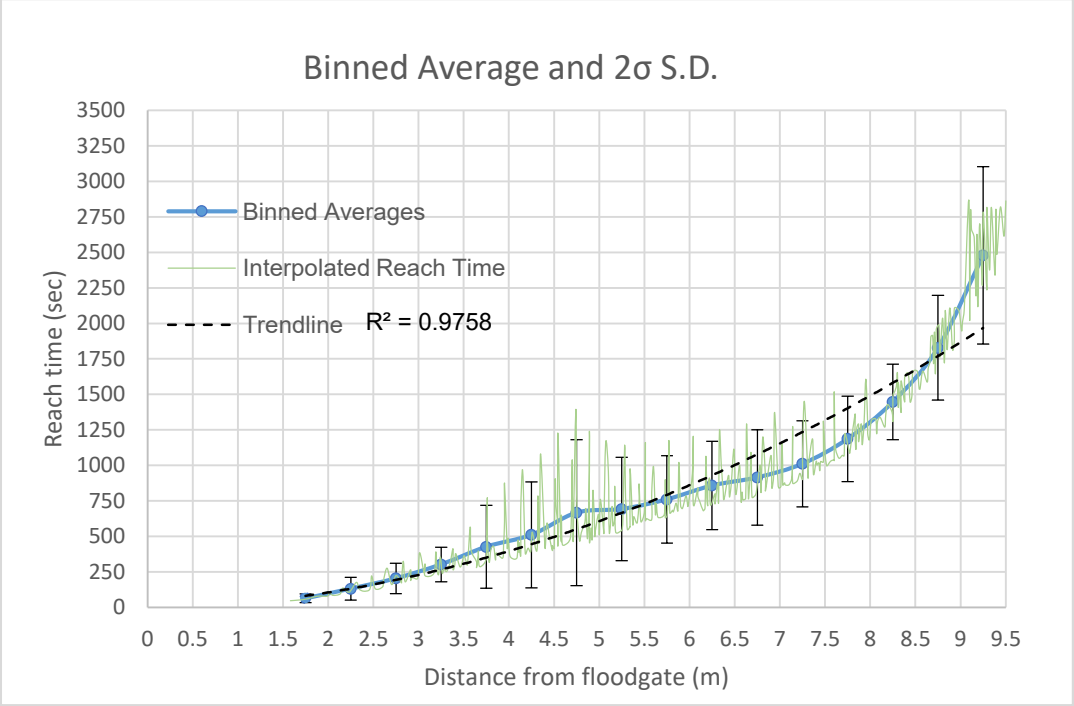

Figure 6. 11: Binned averages μ (blue points) and 2σ standard deviation (vertical bars) overlapped over reach time vs. distance plot (green plot).

The binned average (μ) along with the 2σ standard deviation vertical bars in figure 6.11 cover most of the oscillations of the reach time. However, it is to be noted that in some areas, the μ-2σ gives a significantly lower value than the actual values.

The trend of the plot is very close to a power function (R^2 = 0.9758) of the form $t = Ar^B$ where t is the reach time of the ice piece, r is the distance of the initial position from the floodgate and A,B are constants.

7 CONCLUSION

In this study, the feasibility of extracting ice pieces in a rectangular tank using a single centrally located floodgate is studied. The defined aims of this analysis are to simulate motion of a single ice piece in a water flow generated due to a floodgate, evaluate the feasibility of the method in removing ice piece applicable to the WINICE project, study the effect of the reducing ice sheet length on the motion of the ice piece and finally study the relationship between the time taken for the ice piece to reach the floodgate v/s its initial distance from the floodgate.

The simulation of ice piece motion has been done in two stages: simulating only the water flow in OpenFOAM and later, simulating the ice piece motion in MATLAB based on the flow field from the water flow simulation. In the process of the OpenFOAM simulation, different mesh parameters are obtained such that a balance between simulation accuracy and simulation time is achieved. The simulation of the ice piece is done by calculating the fluid drag forces and the ice-ice friction forces on a rectangular ice piece (of size 1m x 1m x 0.02m) after it is cut from the ice sheet. The acceleration, velocity and the position of the ice piece at different times are then computed based on the force balance equation 5.11.

The simulation of the ice piece motion shows that the floodgate generated flow is able to drag the ice piece from the farthest distance in the tank and takes about 58 minutes (3508 seconds) to do so. The ice pieces near to the floodgate move out quickly and most of the time taken to empty the tank would be spent on the piece farther away from the floodgate. This time can be sped up by increasing the width of the floodgate, thereby increasing the induced flow velocities. Thus, this method for removing ice pieces from the tank is feasible.

On analysing the relationship between the reach time and the distance of the ice piece's initial position from the floodgate, oscillations are found in the plot. These oscillations are due to there being different X and Z coordinates (thus different partial ice sheet length) for the same initial position of the ice piece, causing the simulation to use different interpolated flow field cases for the same distance values. Also, over a distance of 9.5m, there is an unreasonable spike in reach time indicating that beyond a distance of 9.5m, distance is not a valid parameter to judge the reach time. However, despite these oscillations and the unreasonable spikes, there is a visible trend in the reach time vs. distance plot for the majority of the tank domain. The observed trend is of the form $t = Ar^B$ where t is the reach time of the ice piece, r is the distance of the initial position from the floodgate and A,B are constants.

8 FUTURE WORK

Using the fluid simulation in OpenFOAM and subsequent simulation of ice piece motion based on the OpenFOAM generated flow field, the motion of the ice piece in a rectangular tank under the influence of a floodgate is calculated. However, the theoretical calculations done in section 5 are not yet validated experimentally. Aalto University has an ice tank facility which is mainly used for ice model scale tests but can also be used for open water tests. The tank is 40m x 40m x 2.8m in length, width and height respectively. Using this facility, the theoretical calculations of the reach time v/s the distance from the floodgate and also, the effect of the changing ice sheet length needs to be corroborated with experimental data.

Secondly, there have been several assumptions made for the OpenFOAM fluid simulation and also in the calculation of forces on the ice piece due to this fluid flow in section 5.1. In this, several phenomena, such as ice-tank friction, have been neglected. Depending on the errors obtained from the experiments conducted in the Aalto ice tank, these neglected physical phenomena can be added into the existing model if needed. Thereby, making the model more complex. However, the addition of these in the model will depend on significance of the error between the experimental and theoretical values of reach time. The features that have been neglected in the current model (Section 5.1) and can be added in the future work are:

1. **Ice piece rotation:** In this model, only the translation of a single ice piece is taken into consideration. In the future, rotation of the ice piece can also be added in to the model.
2. **Ice-Ice interaction:** Currently, translation of only a single rigid ice piece is taken into consideration. In the future, multiple ice pieces (of arbitrary shapes) can be accounted for simultaneously along with the ice-ice interaction (collision, deformation and cracking).
3. **Ice-tank friction:** In this model, it is assumed that after the collision of the ice piece with the tank walls, the ice piece will slide towards the floodgate frictionlessly. However, in reality, there will be ice to wall friction depending on the material of the tank walls. Along with the sliding movement, the ice will also have a tendency to roll about its center of mass. This can be taken into account in the future.
4. **Change of ice sheet width:** This study only considers the change in the length of the ice sheet and its effect on the reach time of the ice piece. However, as more ice pieces are cut from the ice sheet, the ice sheet dimensions will also change in width. A future study can include the effect of the change width in combination with the changing length of the ice sheet as the ice pieces are cut from the ice sheet.

REFERENCES

1. Rodriguez, M., Luque, S., Alvarez, J. and Coca, J., 2000. A comparative study of reverse osmosis and freeze concentration for the removal of valeric acid from wastewaters. Desalination, 127(1), pp.1-11.

2. Mudd, G.M., 2008. Sustainability reporting and water resources: a preliminary assessment of embodied water and sustainable mining. Mine Water and the Environment, 27(3), pp.136-144.

3. Akcil, A. and Koldas, S., 2006. Acid mine drainage (AMD): causes, treatment and case studies. Journal of Cleaner Production, 14(12), pp.1139-1145.

4. Sörme, L. and Lagerkvist, R., 2002. Sources of heavy metals in urban wastewater in Stockholm. Science of the Total Environment, 298(1), pp.131-145.

5. Lorain, O., Thiebaud, P., Badorc, E. and Aurelle, Y., 2001. Potential of freezing in wastewater treatment: soluble pollutant applications. Water research, 35(2), pp.541-547.

6. Yan, Z.H.A.N.G., Chang-you, L.I., Xiao-yan, Z.H.A.N.G., Xiao-hong, S.H.I. and Wei-ping, L.I., 2011. The Research on Purification Mechanism of Natural Cool Energy and Its Application in Wastewater Treatment. Energy Procedia, 5, pp.2554-2561.

7. Mining Magazine (March 2015), Breaking the Ice, pp. 68-69

8. Bogdan, A., Molina, M.J., Tenhu, H., Bertel, E., Bogdan, N. and Loerting, T., 2014. Visualization of freezing process in situ upon cooling and warming of aqueous solutions. Scientific reports, 4, p.7414.

9. Kurniawan, T.A., Chan, G.Y., Lo, W.H. and Babel, S., 2006. Physico–chemical treatment techniques for wastewater laden with heavy metals. Chemical engineering journal, 118(1), pp.83-98.

10. OpenCFD Ltd. OpenFOAM, The Open Source CFD Toolbox, User Guide, 2015

11. Hirt, C.W. and Nichols, B.D., 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. Journal of computational physics, 39(1), pp.201-225.

12. Ubbink, O., 1997. Numerical prediction of two fluid systems with sharp interfaces (Doctoral dissertation, University of London).

13. Ubbink, H. Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions, Ph.D Thesis, Imperial College of Science, Technology and Medicine, London, 2002.

14. Berberović, E., van Hinsberg, N.P., Jakirlić, S., Roisman, I.V. and Tropea, C., 2009. Drop impact onto a liquid layer of finite thickness: Dynamics of the cavity evolution. Physical Review E, 79(3), p.036306.

15. Deville, M. and Gatski, T.B., 2012. Mathematical modeling for complex fluids and flows. Springer Science & Business Media, p. 155

16. Kestin, J., Sokolov, M. and Wakeham, W.A., 1978. Viscosity of liquid water in the range− 8 C to 150 C. Journal of Physical and Chemical Reference Data, 7(3), pp.941-948.

17. Stockstill, R.L., Daly, S.F. and Hopkins, M.A., 2009. Modeling floating objects at river structures. Journal of Hydraulic Engineering, 135(5), pp.403-414.

18. Hasan, M. and Louhi-Kultanen, M., 2015. Ice growth kinetics modeling of air-cooled layer crystallization from sodium sulfate solutions. Chemical Engineering Science, 133, pp.44-53.

19. Hobbs, P.V., 1974. Ice physics. Oxford university press, pp. 348

20. Clauss, G., Lehmann, E. and Östergaard, C., 2014. Offshore Structures: Volume I: Conceptual Design and Hydromechanics. Springer, pp. 228

21. Schulson, E.M. and Duval, P., 2009. Creep and fracture of ice (Vol. 432). Cambridge: Cambridge University Press, pp. 69.

22. LI, Y.L., ZHU, R.C., MIAO, G.P. and Ju, F.A.N., 2012. Simulation of tank sloshing based on OpenFOAM and coupling with ship motions in time domain. Journal of Hydrodynamics, Ser. B, 24(3), pp.450-457.

23. Cha, J.J. and Wan, D.C., 2011. Numerical wave generation and absorption based on OpenFOAM. Ocean Engineering(Haiyang Gongcheng), 29(3), pp.1-12.

24. Prof. Timo Siikonen, Virtaussimulointi course, Aalto University, Lecture 2, January 2015.

25. BS 5400-2:2006 Steel, concrete and composite bridges: Specification of loads, pp. 20

26. Rusche, H., 2003. Computational fluid dynamics of dispersed two-phase flows at high phase fractions (Doctoral dissertation, Imperial College London (University of London)).

APPENDIX

MATLAB Code for interpolated ice sheet

```
%% INITIALIZE
close all;
clear all;

num_points_x = 20;
num_points_z = 20;
X = linspace(0.501,8.5,num_points_x);
Z = linspace(0.501,4.5,num_points_z);


%Select case and initial ice piece CG position
path =
'C:\Users\vaibh\Desktop\VaibhavThesis\backup\Desktop\Simulation\different
Opening\';
% Case = '1by5\datafiles\Velocity.csv';
gatewidth = 5-(5*(1/5));
endTime = 10000;
dt = 1; %Time step
visual = 0;

%Dimensional parameters of the ice piece
xlen = 1; %length of ice piece in X
zlen = 1; %length of ice piece in Z
iteration = 1;
total_time = 0;
for a=1:1:num_points_x
    folder =
strcat('C:\Users\vaibh\Desktop\VaibhavThesis\backup\Desktop\Simulation\Ic
e piece motion\Interpolated\X_',num2str(X(a)));
    mkdir(folder);
    for b=1:1:num_points_z
        clearvars -except iteration xlen zlen X Z a b visual dt path
endTime gatewidth num_points_x num_points_z final_data
        centerxStart = X(a); %Starting position of the ice piece/CG
        centerzStart = Z(b);
        Xc(1) = centerxStart; %starting location of ice piece CG/cut
position
        Zc(1) = centerzStart;
        iceout = 0;
        Fx = 0; %Total force in X direction
        Fz = 0; %Total force in Z direction
        vx(1) = 0; %Initial X velocity
        vz(1) = 0; %Initial Z velocity
        accelx(1) = 0;
        accelz(1) = 0;
        %Properties of Ice and water
        visc = 1.656e-3; %Viscocity for water between 0 to 4 degrees
ranges from 1.8e-3 to 1.55e-3. So taking the average as 1.656e-3
        Ms = 0.05; % Friction coefficient
        Cd = 1.2; %Drag coefficient
        rho = 1000; %water density
        rhoice = 917; %density of ice
        thck = 0.02; %Total thickness of the ice sheet
        subthck = (rhoice/rho)*thck;%Submerged thickness of the ice sheet
```

```
        Area_b = xlen*zlen; %Area of the bottom face on which shear
stress will act
        m = rhoice * thck * Area_b; %Total mass of the ice piece
        Area_s = subthck*xlen; %Side area of the ice piece

        %% importing velocity files based on partial ice case
        %Deciding the partial ice cases
        icecase = centerxStart/10;
        if(icecase > 0 && icecase <= 0.25)
            Case = 'noIcesheet\datafiles\Velocity.csv';
            U_1 = importdata(strcat(path,Case));
            Case = 'quarterIcesheet\datafiles\Velocity.csv';
            U_2 = importdata(strcat(path,Case));
            U_1 = sortrows(U_1,[1 3]);
            U_2 = sortrows(U_2,[1 3]);
            for c = 1:1:3
                U(:,c) = (U_2(:,c));
            end
            a0 = 0; a1 = 0.25;
            for c = 3:1:6
                U(:,c) = (1-((icecase-a0)/(a1-a0)))*U_1(:,c) + ((icecase-
a0)/(a1-a0))* U_2(:,c);
            end


        end
        if (icecase > 0.25 && icecase <= 0.5)
            Case = 'quarterIcesheet\datafiles\Velocity.csv';
            U_1 = importdata(strcat(path,Case));
            Case = 'halfIcesheet\datafiles\Velocity.csv';
            U_2 = importdata(strcat(path,Case));
            a0 = 0.25; a1 = 0.5;
            U_1 = sortrows(U_1,[1 3]);
            U_2 = sortrows(U_2,[1 3]);
            for c = 1:1:3
                U(:,c) = (U_2(:,c));
            end
            for c = 3:1:6
                U(:,c) = (1-((icecase-a0)/(a1-a0)))*U_1(:,c) + ((icecase-
a0)/(a1-a0))* U_2(:,c);
            end

        end
        if (icecase > 0.5 && icecase <= 0.75)
            Case = 'halfIcesheet\datafiles\Velocity.csv';
            U_1 = importdata(strcat(path,Case));
            Case = 'threefourthIcesheet\datafiles\Velocity.csv';
            U_2 = importdata(strcat(path,Case));
            a0 = 0.5; a1 = 0.75;
            U_1 = sortrows(U_1,[1 3]);
            U_2 = sortrows(U_2,[1 3]);
            for c = 1:1:3
                U(:,c) = (U_2(:,c));
            end
            for c = 3:1:6
                U(:,c) = (1-((icecase-a0)/(a1-a0)))*U_1(:,c) + ((icecase-
a0)/(a1-a0))* U_2(:,c);
            end

        end
```

```
        if (icecase > 0.75 && icecase <= 1)
            Case = 'threefourthIcesheet\datafiles\Velocity.csv';
            U_1 = importdata(strcat(path,Case));
            Case = 'fullIcesheet\datafiles\Velocity.csv';
            U_2 = importdata(strcat(path,Case));
            a0 = 0.75; a1 = 1;
            U_1 = sortrows(U_1,[1 3]);
            U_2 = sortrows(U_2,[1 3]);
            for c = 1:1:3
                U(:,c) = (U_2(:,c));
            end
            for c = 3:1:6
                U(:,c) = (1-((icecase-a0)/(a1-a0)))*U_1(:,c) + ((icecase-
a0)/(a1-a0))* U_2(:,c);
            end

        end


        %% Start simulation
        n = 1;
        for k = 0:dt:endTime
            x1 = Xc(n) - (xlen/2); %Updates limiting parameters after
each timestep that are needed for finding data points within the
rectangular piece.
            x2 = Xc(n) + (xlen/2);
            z1 = Zc(n) - (zlen/2);
            z2 = Zc(n) + (zlen/2);
            %disp(strcat('Time = ',' ',num2str(k)));
            j = 1;
            clear P;
            for i=1:1:length(U(:,1))  %This finds all the data points
lying inside the square ice piece
                if((U(i,1)>=x1 && U(i,1)<=x2) && (U(i,3)>=z1 &&
U(i,3)<=z2)) %X and Z cordinates are within the limits of x1,x2 and z1,z2
                    P(j,1) = U(i,1);
                    P(j,2) = U(i,2);
                    P(j,3) = U(i,3);
                    P(j,4) = U(i,4);
                    P(j,5) = U(i,6);
                    P(j,6) = 1/sqrt(  (U(i,1)-Xc(n))^2 + (U(i,3)-Zc(n))^2
); %Distance from the center. Acts as weight when finding weighted
average of the stress
                    j = j+1;
                end
            end

            Ux(n) = sum(P(:,4).*(P(:,6).^2))/sum(P(:,6).^2); %Finding
water velocity average velocity at the center of the mass
            Uz(n) = sum(P(:,5).*(P(:,6).^2))/sum(P(:,6).^2);
            relvelx = (Ux(n)-vx(n)); %relative water velocity at the CG
            relvelz = (Uz(n)-vz(n));
            Fx = 0.5*Cd*rho*(relvelx^2)*Area_s; %Drag force calculation
            Fz = 0.5*Cd*rho*(relvelz^2)*Area_s;
            %% FOR ICE STILL INSIDE THE ICE SHEET
            if(iceout == 0)
                Fn = Fz; %Normal force for friction force
                Ff = Ms*Fn;
```

```
                Xc(n+1) = Xc(n) + vx(n)*dt; %Calculates the new position
of the center of mass
                Zc(n+1) = Zc(n); %Z remains the same as initial Z since
the ice is only moving in X direction for now.
                accelx(n) = (Fx-Ff)/(m); %Acceleration in the X direction
                vx(n+1) = accelx(n)*dt + vx(n); %Velocity in the next
time step due to accelx
                vz(n+1) = vz(1); %Velocity in the z remains the same as
initial velocity i.e. 0. since the piece has to slide between two other
pieces
                accelz(n) = accelz(1);
                if(Xc(n+1)>9.5)
                    Xc(n+1) = 9.5;
                end
                if(Xc(n+1)<0.5)
                    Xc(n+1) = 0.5;
                end
                if(Zc(n+1)>5)
                    Zc(n+1) = 4.5;
                end
                if(Zc(n+1)<0.5)
                    Zc(n+1) = 0.5;
                end
                if(Xc(n+1)+(xlen/2) >= 10)
                    vx(n+1) = 0;
                    accelx(n) = 0;
                    Fx = 0;
                end
                T(n,1) = k;     %Writing necessary results to a Final
array
                T(n,2) = Xc(n);
                T(n,3) = Zc(n);
                T(n,4) = vx(n);
                T(n,5) = vz(n);
                T(n,6) = accelx(n);
                T(n,7) = accelz(n);
                T(n,8) = Fx;
                T(n,9) = Fz;
                T(n,10) = Ff;

                if((Xc(n+1) - Xc(1) >= xlen)) %checking if the pieces is
still between two other pieces.
                    iceout = 1;  %Flag for checking
                    disp('Ice out of the groove');
                end


                %% After the ice is out of the ice sheet
                % end
            else
                Ff = 0;
                Xc(n+1) = Xc(n) + vx(n)*dt; %Calculates the new position
of the center of mass
                Zc(n+1) = Zc(n) + vz(n)*dt; %Z remains the same as
initial Z since the ice is only moving in X direction for now.
                accelx(n) = (Fx)/(m); %Acceleration in the X direction
                vx(n+1) = (accelx(n)*dt) + vx(n);
                accelz(n) = (Fz)/(m);
                vz(n+1) = (accelz(n)*dt) + vz(n);
```

```
                if(Xc(n+1)>9.5)
                    Xc(n+1) = 9.5;
                end
                if(Xc(n+1)<0.5)
                    Xc(n+1) = 0.5;
                end
                if(Zc(n+1)>5)
                    Zc(n+1) = 4.5;
                end
                if(Zc(n+1)<0.5)
                    Zc(n+1) = 0.5;
                end
                if(Xc(n+1)+(xlen/2) >= 10)
                    vx(n+1) = 0;
                    accelx(n) = 0;
                    Fx = 0;
                end

                T(n,1) = k;
                T(n,2) = Xc(n);
                T(n,3) = Zc(n);
                T(n,4) = vx(n);
                T(n,5) = vz(n);
                T(n,6) = accelx(n);
                T(n,7) = accelz(n);
                T(n,8) = Fx;
                T(n,9) = Fz;
                T(n,10) = Ff;

            end

            if((Xc(n+1)+(xlen/2) >= 10) && (Zc(n+1)>= gatewidth)) % If
the block hits the wall near the gate or hits the symmtery plane
                break;
            end
            n = n+1;  % n is used to as index to read and write neeeded
values.


        end

        disp(strcat('Start X = ','  ',num2str(centerxStart),' seconds'));
        disp(strcat('Start Z = ','  ',num2str(centerzStart),' seconds'));
        disp(strcat('Final time = ','  ',num2str(T(end,1)),' seconds'));
        %total_time = total_time + T(end,1);
        %disp(strcat('Total elapsed = ','  ',num2str(total_time),'
seconds'));
        %% DISPLAY RESULT
        if(visual==1)
            disp(strcat('Total time = ','  ',num2str(T(end,1)),'
seconds'));

            n1 = 1;
            for t = 0:dt:endTime
                if(n1<=size(T,1))
                    rectangle('Position',[0 0 10 5])
                    x1 = T(n1,2)-(xlen/2);
```

```
                    z1 = T(n1,3)-(zlen/2);
                    rectangle('Position',[x1 z1 xlen
zlen],'FaceColor','green');
                    rectangle('Position',[0 0 (T(1,2)+xlen/2) (T(1,3)-
zlen/2)],'FaceColor','blue','lineStyle','none');
                    rectangle('Position',[0 (T(1,3)+zlen/2)
(T(1,2)+xlen/2) (5-T(1,3)-
zlen/2)],'FaceColor','blue','lineStyle','none');
                    rectangle('Position',[0 (T(1,3)-zlen/2) (T(1,2)-
xlen/2) zlen],'FaceColor','blue','lineStyle','none');
                    line([10 10],[5
gatewidth],'Color','r','lineWidth',5);
                    title(strcat('Time = ',num2str(T(n1,1))));
                    grid on;
                    xlabel 'X direction';
                    ylabel 'Z direction';
                    axis([0 10 0 5]);
                    n1 = n1+(10/dt);
                    pause(0.01);
                else break;

                end
            end
        end
        final_data(iteration,1) = centerxStart;
        final_data(iteration,2) = centerzStart;
        final_data(iteration,3) = T(end,1);
        final_data(iteration,4) = sqrt(((centerxStart - 10)^2) +
((centerzStart - 5)^2));
        iteration = iteration + 1;

        Tfile =  strcat('Z_',num2str(centerzStart),'.txt');
        fullFileName =
fullfile(strcat('C:\Users\vaibh\Desktop\VaibhavThesis\backup\Desktop\Simu
lation\Ice piece motion\Interpolated\X_',num2str(centerxStart)),Tfile);
        dlmwrite(fullFileName,T);
    end
end
%%
dlmwrite('Final_Data.csv', final_data);
times = transpose(vec2mat(final_data(:,3),num_points_z));
fig = surf(X,Z,times);
caxis([0 10000]);
view(2);
shading interp;
c = colorbar;
c.Label.String = 'Reach time(seconds)';
xlabel 'X (m)';
ylabel 'Z (m)';
xlim([0 10]);
ylim([0 5]);
saveas(fig,'Interpolated ice sheet time contour.png');
```

MATLAB code for individual partial ice case

The code is the same for each partial ice case except for the velocity flow field file imported to MATLAB. In the code below, the only difference between different cases is the variable '**Case**'. The code below is for the 100% ice sheet case.

```
%% INITIALIZE
close all;
clear all;

num_points_x = 20;
num_points_z = 20;
X = linspace(0.501,8.5,num_points_x);
Z = linspace(0.501,4.5,num_points_z);

%Select case and initial ice piece CG position
path =
'C:\Users\vaibh\Desktop\VaibhavThesis\backup\Desktop\Simulation\different
Opening\';
% Case = '1by5\datafiles\Velocity.csv';
gatewidth = 5-(5*(1/5));
endTime = 20000;
dt = 1; %Time step
visual =0;

%Initialization of forces and velocities



%Dimensional parameters of the ice piece
xlen = 1; %length of ice piece in X
zlen = 1; %length of ice piece in Z
iteration = 1;
for a=1:1:num_points_x
    folder =
strcat('C:\Users\vaibh\Desktop\VaibhavThesis\backup\Desktop\Simulation\Ic
e piece motion\Only fullIcesheet\X_',num2str(X(a)));
    mkdir(folder);
    for b=1:1:num_points_z
        clearvars -except iteration xlen zlen X Z a b visual dt path
endTime gatewidth num_points_x num_points_z final_data
        centerxStart = X(a); %Starting position of the ice piece/CG
        centerzStart = Z(b);
        Xc(1) = centerxStart; %starting location of ice piece CG/cut
position
        Zc(1) = centerzStart;
        iceout = 0;
        Fx = 0; %Total force in X direction
        Fz = 0; %Total force in Z direction
        vx(1) = 0; %Initial X velocity
        vz(1) = 0; %Initial Z velocity
        accelx(1) = 0;
        accelz(1) = 0;
        %Properties of Ice and water
        visc = 1.656e-3; %Viscocity for water between 0 to 4 degrees
ranges from 1.8e-3 to 1.55e-3. So taking the average as 1.656e-3
        Ms = 0.05; % Friction coefficient
        Cd = 1.2; %Drag coefficient
        rho = 1000; %water density
```

```
        rhoice = 917; %density of ice
        thck = 0.02; %Total thickness of the ice sheet
        subthck = (rhoice/rho)*thck;%Submerged thickness of the ice sheet
        Area_b = xlen*zlen; %Area of the bottom face on which shear
stress will act
        m = rhoice * thck * Area_b; %Total mass of the ice piece
        Area_s = subthck*xlen;

        %% importing velocity files based on partial ice case

        Case = 'fullIcesheet\datafiles\Velocity.csv';
        U = importdata(strcat(path,Case));

        %% Start simulation
        n = 1;
        for k = 0:dt:endTime
            x1 = Xc(n) - (xlen/2); %Updates limiting parameters after
each timestep that are needed for finding data points within the
rectangular piece.
            x2 = Xc(n) + (xlen/2);
            z1 = Zc(n) - (zlen/2);
            z2 = Zc(n) + (zlen/2);
            %disp(strcat('Time = ',' ',num2str(k)));
            j = 1;
            clear P;
            for i=1:1:length(U(:,1))  %This finds all the data points
lying inside the square ice piece
                if((U(i,1)>=x1 && U(i,1)<=x2) && (U(i,3)>=z1 &&
U(i,3)<=z2)) %X and Z cordinates are within the limits of x1,x2 and z1,z2
                    P(j,1) = U(i,1);
                    P(j,2) = U(i,2);
                    P(j,3) = U(i,3);
                    P(j,4) = U(i,4);
                    P(j,5) = U(i,6);
                    P(j,6) = 1/sqrt(  (U(i,1)-Xc(n))^2 + (U(i,3)-Zc(n))^2
); %Distance from the center. Acts as weight when finding weighted
average of the stress
                    j = j+1;
                end
            end

            Ux(n) = sum(P(:,4).*(P(:,6).^2))/sum(P(:,6).^2); %Finding
water velocity average velocity at the center of the mass
            Uz(n) = sum(P(:,5).*(P(:,6).^2))/sum(P(:,6).^2);
            relvelx = (Ux(n)-vx(n)); %relative water velocity at the CG
            relvelz = (Uz(n)-vz(n));
            Fx = 0.5*Cd*rho*(relvelx^2)*Area_s; %Drag force calculation
            Fz = 0.5*Cd*rho*(relvelz^2)*Area_s;
            %% FOR ICE STILL INSIDE THE ICE SHEET
            if(iceout == 0)
                Fn = Fz; %Normal force for friction force
                Ff = Ms*Fn;
                Xc(n+1) = Xc(n) + vx(n)*dt; %Calculates the new position
of the center of mass
                Zc(n+1) = Zc(n); %Z remains the same as initial Z since
the ice is only moving in X direction for now.
                accelx(n) = (Fx-Ff)/(m); %Acceleration in the X direction
                vx(n+1) = accelx(n)*dt + vx(n); %Velocity in the next
time step due to accelx
```

```
                vz(n+1) = vz(1); %Velocity in the z remains the same as
initial velocity i.e. 0. since the piece has to slide between two other
pieces
                accelz(n) = accelz(1);
                if(Xc(n+1)>9.5)
                    Xc(n+1) = 9.5;
                end
                if(Xc(n+1)<0.5)
                    Xc(n+1) = 0.5;
                end
                if(Zc(n+1)>5)
                    Zc(n+1) = 4.5;
                end
                if(Zc(n+1)<0.5)
                    Zc(n+1) = 0.5;
                end
                if(Xc(n+1)+(xlen/2) >= 10)
                    vx(n+1) = 0;
                    accelx(n) = 0;
                    Fx = 0;
                end
                if(Zc(n+1)+(zlen/2) >= 5 || Zc(n+1)-(zlen/2) <= 0 )
                    vz(n+1) = 0;
                    accelz(n) = 0;
                    Fz = 0;
                end

                T(n,1) = k;     %Writing necessary results to a Final
array
                T(n,2) = Xc(n);
                T(n,3) = Zc(n);
                T(n,4) = vx(n);
                T(n,5) = vz(n);
                T(n,6) = accelx(n);
                T(n,7) = accelz(n);
                T(n,8) = Fx;
                T(n,9) = Fz;
                T(n,10) = Ff;

                if((Xc(n+1) - Xc(1) >= xlen)) %checking if the pieces is
still between two other pieces.
                    iceout = 1;  %Flag for checking
                    disp('Ice out of the groove');
                end


                %% After the ice is out of the ice sheet
                % end
            else
                Ff = 0;
                Xc(n+1) = Xc(n) + vx(n)*dt; %Calculates the new position
of the center of mass
                Zc(n+1) = Zc(n) + vz(n)*dt; %Z remains the same as
initial Z since the ice is only moving in X direction for now.
                accelx(n) = (Fx)/(m); %Acceleration in the X direction
                vx(n+1) = (accelx(n)*dt) + vx(n);
                accelz(n) = (Fz)/(m);
                vz(n+1) = (accelz(n)*dt) + vz(n);
```

```
                if(Xc(n+1)>9.5)
                    Xc(n+1) = 9.5;
                end
                if(Xc(n+1)<0.5)
                    Xc(n+1) = 0.5;
                end
                if(Zc(n+1)>5)
                    Zc(n+1) = 4.5;
                end
                if(Zc(n+1)<0.5)
                    Zc(n+1) = 0.5;
                end
                if(Xc(n+1)+(xlen/2) >= 10)
                    vx(n+1) = 0;
                    accelx(n) = 0;
                    Fx = 0;
                end
                if(Zc(n+1)+(zlen/2) >= 5 || Zc(n+1)-(zlen/2) <= 0 )
                    vz(n+1) = 0;
                    accelz(n) = 0;
                    Fz = 0;
                end

                T(n,1) = k;
                T(n,2) = Xc(n);
                T(n,3) = Zc(n);
                T(n,4) = vx(n);
                T(n,5) = vz(n);
                T(n,6) = accelx(n);
                T(n,7) = accelz(n);
                T(n,8) = Fx;
                T(n,9) = Fz;
                T(n,10) = Ff;

            end

            if((Xc(n+1)+(xlen/2) >= 10) && (Zc(n+1)>= gatewidth)) % If
the block hits the wall near the gate or hits the symmtery plane
                break;
            end
            n = n+1;  % n is used to as index to read and write neeeded
values.


        end

        disp(strcat('Start X = ','  ',num2str(centerxStart),' m'));
        disp(strcat('Start Z = ','  ',num2str(centerzStart),' m'));
        disp(strcat('Total time = ','  ',num2str(T(end,1)),' seconds'));
        %% DISPLAY RESULT
        if(visual==1)
            disp(strcat('Total time = ','  ',num2str(T(end,1)),'
seconds'));

            n1 = 1;
            for t = 0:dt:endTime
                if(n1<=size(T,1))
                    rectangle('Position',[0 0 10 5])
                    x1 = T(n1,2)-(xlen/2);
```

```
                    z1 = T(n1,3)-(zlen/2);
                    rectangle('Position',[x1 z1 xlen
zlen],'FaceColor','green');
                    rectangle('Position',[0 0 (T(1,2)+xlen/2) (T(1,3)-
zlen/2)],'FaceColor','blue','lineStyle','none');
                    rectangle('Position',[0 (T(1,3)+zlen/2)
(T(1,2)+xlen/2) (5-T(1,3)-
zlen/2)],'FaceColor','blue','lineStyle','none');
                    rectangle('Position',[0 (T(1,3)-zlen/2) (T(1,2)-
xlen/2) zlen],'FaceColor','blue','lineStyle','none');
                    line([10 10],[5
gatewidth],'Color','r','lineWidth',5);
                    title(strcat('Time = ',num2str(T(n1,1))));
                    grid on;
                    xlabel 'X direction';
                    ylabel 'Z direction';
                    axis([0 10 0 5]);
                    n1 = n1+(10/dt);
                    pause(0.01);
                else break;

                end
            end
        end
        final_data(iteration,1) = centerxStart;
        final_data(iteration,2) = centerzStart;
        final_data(iteration,3) = T(end,1);
        final_data(iteration,4) = sqrt(((centerxStart - 10)^2) +
((centerzStart - 5)^2));
        iteration = iteration + 1;

        Tfile =  strcat('Z_',num2str(centerzStart),'.txt');
        fullFileName =
fullfile(strcat('C:\Users\vaibh\Desktop\VaibhavThesis\backup\Desktop\Simu
lation\Ice piece motion\Only
fullIcesheet\X_',num2str(centerxStart)),Tfile);
        dlmwrite(fullFileName,T);

        % close all;
    end


end
%%
dlmwrite('Final_Data.csv', final_data);
times = transpose(vec2mat(final_data(:,3),num_points_z));
fig = surf(X,Z,times);
caxis([0 10000]);
view(2);
shading interp;
c = colorbar;
c.Label.String = 'Reach time(seconds)';
xlabel 'X (m)';
ylabel 'Z (m)';
xlim([0 10]);
ylim([0 5]);
saveas(fig,'100% ice sheet time contour.png');
```